

IMPROVING THE COMPUTATIONAL EFFICIENCY OF SUBSPACE
ALGORITHMS FOR FREQUENCY ESTIMATION
OF SINUSOIDAL SIGNALS

By
THOMAS HENRY WALLACE

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1994

Copyright 1994 by Thomas Henry Wallace

ACKNOWLEDGEMENTS

I would like to thank all the members of my committee, especially my chairman Dr. Fred Taylor, whose style of supervision was perfectly suited to my style of investigation. Thoughtful discussions on frequency estimation, with Dr. John M. M. Anderson, and Toeplitz eigenanalysis, with Dr. Kermit Sigmon, also contributed greatly to the organization and content of those sections of this work.

My friends and colleagues at ARCO Power Technologies have been extremely helpful, especially Drs. Dave MacEnanny and Bob Short, with whom I had many interesting and entertaining conversations. The financial support provided by ARCO Power Technologies, and its president, Dr. Ramy Shanny, throughout my time at the University of Florida is gratefully acknowledged.

Finally, I would like to thank Drs. T. F. Chan of UCLA and P. C. Hansen of the Technical University of Denmark, for providing the source code for their look-ahead Levinson algorithm, and Dr. W. F. Trench of Trinity University for supplying an implementation of his original fast Toeplitz eigensolver for comparison.

TABLE OF CONTENTS

	<u>page</u>
ACKNOWLEDGEMENTS	ii
ABSTRACT	v
CHAPTERS	
1 INTRODUCTION	1
2 SIGNAL MODELS AND STATISTICAL LIMITS ON ESTIMATION	5
Modeling the Signal	5
Statistical Bounds	10
Derivation of Bounds	17
Comparison of Bounds	30
3 SUBSPACE ESTIMATION TECHNIQUES	39
Eigendecomposition of Exact Autocorrelation Matrices	39
Eigendecomposition of Estimated Autocorrelation Matrices	44
Estimating the Number of Signals	45
Subspace Methods	49
Computational Considerations	57
4 AUTOCORRELATION ESTIMATES FOR SUBSPACE METHODS	62
The Temporal Autocorrelation Matrix	64
Autocorrelation Matrix Estimates	67
Comparison of Autocorrelation Estimators	71
Effects of Errors in Autocorrelation Estimation	76
Comparison of Frequency Estimates	82
5 FAST ALGORITHMS FOR SOLVING TOEPLITZ EQUATIONS	93
Solving the Yule-Walker Equation	93
General Toeplitz Systems	105
Numerical Properties of Toeplitz Algorithms	108
6 FAST ALGORITHMS FOR TOEPLITZ EIGENDECOMPOSITION	112
The Standard Eigenproblem	113
Fast Computation of Any Eigenvalue and Eigenvector	118
Bracketing the Desired Eigenvalue	118
Locating the Roots of $E(\lambda)$	125
The Chan-Hansen Algorithm: A Stable Method	131
The Generalized Eigenproblem	132

7	VERIFYING THE ACCURACY OF RESULTS	135
	An Extremely Simple Test	136
	Error Detection	136
	Comparing a Toeplitz Estimate with the Covariance Estimate	146
	Frequency Estimation Accuracy	147
	Verifying Generalized Eigendecompositions	149
8	ESTIMATING THE NUMBER OF SIGNALS	150
	Bounding the Number of Signals Estimate	151
	Computing Bounds for a Single Value of n	152
	Narrowing the Bracket Intervals	156
	Computing Bounds for Consecutive Values of n	157
9	COMPARISON OF CONVENTIONAL AND FAST METHODS . .	158
	Implementation of the ESPRIT Algorithm	159
	Simulation Results	163
10	CONCLUSIONS	181
	A Critical Review	181
	Areas for Further Research	183
APPENDICES		
A	REVIEW OF MATRIX ALGEBRA	189
	Norms	189
	Special Matrices	190
	Matrix Eigendecomposition	191
	Generalized Eigendecomposition	192
B	ROUTINES FOR FAST SOLUTION OF TOEPLITZ SYSTEMS . .	194
	Introduction	194
	Listings	195
REFERENCES		204
BIOGRAPHICAL SKETCH		213

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

IMPROVING THE COMPUTATIONAL EFFICIENCY OF SUBSPACE
ALGORITHMS FOR FREQUENCY ESTIMATION
OF SINUSOIDAL SIGNALS

By

Thomas Henry Wallace

August, 1994

Chairman: Dr. Fred J. Taylor

Major Department: Electrical Engineering

The principal focus of this work is the application of fast techniques for Toeplitz eigendecomposition to improve the computational efficiency of the subspace algorithms for frequency estimation of sinusoidal signals. Two forms of fast Toeplitz eigendecomposition algorithms are developed: the first is an improved version of an earlier fast Toeplitz algorithm that computes any eigenvalue and the associated eigenvector of a real symmetric $M \times M$ Toeplitz matrix in $\mathcal{O}(M^2)$ floating-point operations; the second employs the recently developed “superfast” techniques for solving Toeplitz systems to compute any eigenvalue and the associated eigenvector in $\mathcal{O}(M \log^2 M)$ operations, a lower asymptotic complexity than any previous method.

A principal concern in the use of these fast and superfast algorithms is their numerical instability, which may lead to inaccurate results in rare cases. In this work, reliable, efficient tests are described for assuring the accuracy of the computed eigenvalues and eigenvectors. These tests are based on residual bounds, and employ

fast algorithms for Toeplitz matrix-vector multiplication, making their use practical even with the superfast eigendecomposition algorithms.

To provide a benchmark for evaluating frequency estimates computed by the new techniques, a new bound for the estimation of the frequencies of sinusoidal signals is derived. The Bhattacharyya bound, an extension of the well-known Cramér-Rao bound, is a lower bound to the variance of any unbiased estimator of the frequencies of sinusoidal signals. This bound is the tightest known bound for this problem, and is significantly tighter than the Cramér-Rao bound in cases where the signal-to-noise ratio is low, the number of samples is small, or the sinusoids are closely spaced in frequency.

Finally, the new techniques for Toeplitz eigendecomposition are used to produce a more efficient variant of the ESPRIT algorithm for frequency estimation. The performance of this fast variant and the original ESPRIT algorithm are compared for a wide variety of situations, and it is shown that in many cases, the fast algorithms can produce a more accurate estimate with less computation, by employing a larger Toeplitz estimate of the autocorrelation matrix, rather than the covariance estimate used in the conventional implementations. In cases where long data records are available, this fast Toeplitz ESPRIT can produce more accurate frequency estimates with less computation than the standard ESPRIT algorithm.

CHAPTER 1 INTRODUCTION

In many signal processing applications, it is necessary to determine the frequencies of sinusoidal signals from a data record that is corrupted by additive noise. Estimating the parameters of sinusoidal signals in noise is one of the oldest problems in the field of signal processing, dating back at least as far as the 18th century work of Kelvin and Schuster on analysis and prediction of tides and temperatures. Problems of this type occur not only in analysis of time series data, but also in array bearing estimation, that is, the estimation of the directions of arrival of signals from data gathered by a sensor array.

The first techniques to be applied to this problem were the classical spectral estimators, such as the periodogram, which are based on the Fourier transform. These estimators provide good performance for both high and low signal-to-noise ratios, and are simple to compute. If an $M \times M$ autocorrelation matrix is used, for large M the classical techniques require a number of operations proportional to $M \log M$; generally, the notation $\mathcal{O}(M \log M)$ will be used to indicate the asymptotic complexity of algorithms as the problem size becomes large. The principal shortcoming of the classical estimators is their low resolution, that is, their inability to separate signals with small frequency differences. Later, techniques based on modeling the signal as an autoregressive process were developed. These approaches have much higher resolution than classical techniques, and at $\mathcal{O}(M^2)$, require only a modest amount of computation. However, their performance deteriorates rapidly as the signal-to-noise ratio decreases, and they are actually inferior to classical techniques at low signal-to-noise ratios.

In the past 20 years, a class of estimators based on concepts drawn from linear algebra has been developed. These techniques, commonly referred to as subspace methods, include the Pisarenko harmonic decomposition, MUSIC, the principal components method, and ESPRIT. They combine high resolution with good performance at low signal-to-noise ratios, and are among the most accurate methods known for frequency estimation. The principal disadvantage of the subspace methods is their computational burden: all of the widely used subspace techniques perform an eigen-decomposition of an estimate of the autocorrelation matrix, which requires $\mathcal{O}(M^3)$ operations when computed by conventional methods. If M is large, the difference between $\mathcal{O}(M^3)$ and the $\mathcal{O}(M \log M)$ of the classical methods, or the $\mathcal{O}(M^2)$ of autoregressive modeling, can be prohibitive.

Since the computational requirements of all of the techniques used for estimating frequencies increase with M , it is tempting to reduce the length of the autocorrelation estimate in order to reduce the computational load. Unfortunately, a basic result in the estimation theory of sinusoidal signals is that the best achievable variance for an unbiased frequency estimate improves as the cube of the number of samples. In practice, this means that reducing M degrades the accuracy of the estimate severely. If accurate frequency estimates are required, M must be as large as is practical.

Frequency estimation problems therefore present a difficult choice between the subspace methods, which offer high resolution and good performance at all signal-to-noise ratios, but require a large, possibly prohibitive amount of computation, and autoregressive or classical methods, which are less demanding computationally but have important drawbacks such as poor performance at low signal-to-noise ratios or lower resolution.

Recently, methods have been developed that greatly reduce the computation required to find eigenvalues and eigenvectors of Toeplitz matrices. These methods

employ well-known techniques for solving Toeplitz systems, such as the Levinson-Durbin algorithm, to compute an eigenvalue and the associated eigenvector in $\mathcal{O}(M^2)$ operations. By computing a Toeplitz estimate of the autocorrelation matrix, and employing these methods to compute its eigendecomposition, the efficiency of the subspace techniques can be dramatically improved. The principal goal of this work is to develop these techniques and to characterize their performance, in terms of both speed and accuracy, in comparison to standard implementations of the subspace techniques.

To provide a means for evaluating the performance of both the standard and the new techniques, an analysis of statistical bounds on the estimation of frequencies of sinusoids has been performed. In addition to a discussion of the well-known Cramér-Rao bound, an extended version of this bound, called the Bhattacharyya bound, is derived for the first time. This result provides a tighter bound on the variance of estimators of sinusoidal frequencies, and is much tighter than the Cramér-Rao bound in cases of low signal-to-noise ratios, short data records, or signals whose frequencies are very closely spaced. The new bound substantially narrows the gap between the performance of existing estimators and the tightest known bounds, and may be a first step toward an understanding of the limits on the performance of frequency estimators.

Previous fast Toeplitz eigensolvers were $\mathcal{O}(M^2)$, and questions existed concerning their numerical stability. In the present work, the efficiency of earlier fast Toeplitz eigensolvers has been further improved by employing recently developed “superfast” algorithms for solving Toeplitz systems, which reduce the complexity of the eigen-decomposition. The resulting algorithm computes an eigenvalue and the associated eigenvector of a Toeplitz matrix in $\mathcal{O}(M \log^2 M)$ operations; this is the lowest asymptotic complexity of any known algorithm for the Toeplitz eigenproblem. To address

the issue of numerical stability, a set of simple, efficient tests for verifying the accuracy of an eigendecomposition of a Toeplitz matrix are presented. These tests are reliable, useful for both the earlier fast techniques and the new superfast algorithms, and have negligible computational cost.

Finally, the fast Toeplitz eigendecomposition techniques have been used to develop a fast variant of the ESPRIT algorithm for frequency estimation. The performance of the fast method has been compared to a conventional implementation, and it has been shown that the fast method can often produce more accurate frequency estimates with less computation. Although the use of a Toeplitz estimate degrades the accuracy of the estimate somewhat, the fast Toeplitz algorithms make it possible to use much larger autocorrelation matrices. In cases where long data records are available, or when the signal-to-noise ratio is low, the increased accuracy due to the use of a larger matrix more than compensates for the loss due to the use of a Toeplitz estimate.

CHAPTER 2

SIGNAL MODELS AND STATISTICAL LIMITS ON ESTIMATION

The fundamental goal of any estimation technique is to produce an accurate estimate \hat{x} of an unknown parameter x ; in this chapter, we will consider limits on the accuracy of estimation of sinusoidal frequencies. The accuracy of an estimator is typically defined in terms of its bias and variance: an ideal estimation technique would have zero bias and the smallest possible variance. It is not immediately apparent that there is any lower bound to the variance, however, the performance of any estimator of a random quantity is limited by statistical bounds.

Statistical bounds are derived from a model of the random process that is assumed to have produced the received signal, and apply only when this assumption is valid. Many techniques exist for bounding the performance of estimators; in this chapter, the Cramér-Rao and Bhattacharyya bounds for estimation of the frequencies of real sinusoids in additive white Gaussian noise are derived. Both these bounds set lower limits to the variance of any unbiased estimate of a parameter of the random process, providing a useful benchmark for evaluating the performance of estimation techniques; although the Cramér-Rao bounds have been derived previously [1], the Bhattacharyya bounds are a new result. To calculate either of these bounds, it is first necessary to define the random process that is assumed to be the source of the signal.

Modeling the Signal

The frequency estimation problem may be simply stated: given a set of observed data $y[k]$, containing L points, and assuming that the data consist of sinusoids in additive noise, estimate the frequencies of the sinusoids. The problem may also be

formulated in the following equivalent manner: given a data record, find the frequencies of a sinusoidal model that provides the best fit (in some appropriate measure) to the data. We will briefly review the two most widely used models for the signals encountered in frequency estimation: the deterministic and stochastic signal models.

The Deterministic Model

A signal composed of sinusoids in stationary noise may be modeled as the output of a random process that is parameterized by the amplitudes, frequencies, and phases of the sinusoids, and by the variance and autocorrelation of the noise. The underlying signal whose parameters are to be estimated is modeled as a real-valued signal $\tilde{x}[k]$ consisting of N sinusoids:

$$\tilde{x}[k] = \sum_{i=1}^N a_i \cos(\omega_i k + \phi_i). \quad (2.1)$$

It will be assumed that there are exactly N distinct sinusoids (that is, $a_i > 0$ and $\omega_i \neq \omega_j$ if $i \neq j$), that the frequencies are below the Nyquist limit ($0 < \omega_i < \pi$), and that the phases are unambiguously specified ($-\pi < \phi_i \leq \pi$), for all $i \in \{1 \dots N\}$. The probabilistic model for the observed data $y[k]$ is the noisy signal model $\tilde{y}[k]$:

$$\tilde{y}[k] = \sum_{i=1}^N a_i \cos(\omega_i k + \phi_i) + n[k], \quad (2.2)$$

where $n[k]$ is the noise. For convenience in the derivation, the observed data $y[k]$, $k \in \{k_0 + 1, \dots, k_0 + L\}$, where k_0 is the offset of the start of the data record, may be grouped into a vector \mathbf{y} ; similarly, the model signals $\tilde{x}[k]$ and $\tilde{y}[k]$ may be referred to as $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. Although the noisy signal model is a random process, the underlying noise-free signal $\tilde{\mathbf{x}}$ is deterministic, and it is from this fact that the model takes its name. The deterministic signal model is a nonstationary, non-Gaussian random process.

Given the model \tilde{y} , and the set of observed data y , the problem is to determine the parameters of \tilde{y} so that if the data are the output of a process like that assumed in the model, the parameters of \tilde{y} are good estimates of the parameters of the observed data. We will assume that it is known a priori that the use of the sinusoidal model is appropriate.

In the following derivations, we will take $n[k]$ to be a zero-mean white Gaussian noise process with variance σ^2 . This form of the deterministic model has been essentially the only one treated, because of the complexity of the derivation for colored or non-Gaussian noise. Bounds for the case of colored noise may be derived by following the same steps with the appropriate joint probability density function for the noise samples $n[k]$, although the algebraic complications are formidable. For non-Gaussian noise whose probability density function $p(w)$ is exponential, the likelihood function is also exponential, and a procedure similar to that given below may be followed to yield the desired bounds. Non-Gaussian noise with a non-exponential density function introduces additional complications to the derivation, since expressions of the form $\frac{\partial^n P(\Theta)}{\partial \Theta^n}/P(\Theta)$, where $P(\Theta)$ is the likelihood function defined below, arise in the derivation of bounds. If $p(w)$ is not exponential, these expressions no longer have a simple form.

For the white noise case, the parameters of the model $\tilde{y}[k]$ are the amplitudes a_i , the frequencies ω_i , and the phases ϕ_i of the sinusoids, and the variance σ^2 of the noise. The complete set of parameters may be arranged into a vector Θ :

$$\Theta \equiv [\sigma^2, a_1, \omega_1, \phi_1, \dots, a_N, \omega_N, \phi_N]^T \quad (2.3)$$

If the parameters Θ of the model and the observed data \mathbf{y} are given, the probability density of $\tilde{\mathbf{y}}$ is given by

$$p(\tilde{\mathbf{y}}|\Theta, \mathbf{y}) = \frac{1}{(2\pi\sigma^2)^{L/2}} \prod_{k=k_0+1}^{k_0+L} \exp(-(y[k] - \tilde{x}[k])^2/2\sigma^2) \quad (2.4)$$

If the signal vector \mathbf{y} is fixed, and the parameters Θ of the model are considered as variables, then the expression above is referred to as the likelihood function of the parameters Θ . To indicate whether an expression is employed as a probability density function or a likelihood, the notation $p(\tilde{\mathbf{y}}|\Theta, \mathbf{y})$ will be used for density functions, and $P(\Theta)$ for the corresponding likelihood functions.

Maximum Likelihood Estimation

One method for estimating the parameters of a signal is simply to maximize the likelihood function $P(\Theta)$ for the observed data; this is maximum likelihood estimation. Maximum likelihood estimators have several attractive properties, including consistency (as the number of samples approaches infinity, the estimate approaches the true value) and asymptotic efficiency (as the number of samples approaches infinity, the variance of the estimate approaches the minimum possible variance for an unbiased estimator, which is the Cramér-Rao bound). These two attractive properties, however, hold only for infinite data records. One property that does hold for finite records is that if any estimator is efficient (i.e., achieves the Cramér-Rao bound) for a finite number of samples, then the maximum likelihood approach will find an efficient estimator.

In practice, we are most concerned with the performance of an estimator on finite data records; for example, the subspace techniques' principal advantage over the much simpler classical approaches is that subspace techniques can resolve signals closely spaced in frequency using fewer samples of the data. The attractive properties

of the maximum likelihood estimator are for the most part asymptotic properties, and since it has been shown that the maximum likelihood estimator is not efficient for finite sample lengths when noise is present [2], the maximum likelihood estimator is not necessarily superior to any other for this problem.

One major barrier to the use of the maximum likelihood estimator for frequency estimation is that the maximum of $P(\Theta)$ is very difficult to find. The likelihood function is a very complex function, with many local maxima [3], and to be certain of locating the global maximum, it is necessary to have a very good estimate of the frequencies as a starting point for numerical maximization. In fact, subspace methods are often used to generate the starting points for maximum likelihood estimators.

The Stochastic Model

As we will see later in this chapter, the deterministic signal model is relatively detailed, and calculations based on this model often become impractically complex. In those cases, a simpler model, referred to as the stochastic signal model, may be used in place of the deterministic model to make the analysis more tractable. The stochastic model assumes the signal is a stationary zero-mean Gaussian random process, parameterized by its (Toeplitz) autocorrelation matrix Σ ; the likelihood function of this model is given by

$$P_s(\tilde{\mathbf{y}}|\Sigma, \mathbf{y}) = \frac{1}{(2\pi \det(\Sigma))^{1/2}} \exp(-\mathbf{y}^T \Sigma^{-1} \mathbf{y}/2).$$

This appears to be a major departure from the deterministic signal model; however, the two models are in fact related in a simple way by the assumptions made about the coefficients of the sinusoidal model. In the deterministic model, all the parameters of the signal were assumed to be (unknown) constants. If the amplitudes in a sinusoidal signal model are taken instead to be Rayleigh distributed random

variables, and the phases to be uniformly randomly distributed between $-\pi$ and π , independent of the amplitudes, then the output of the sinusoidal model is zero-mean, stationary, and Gaussian [4, p. 48], exactly as assumed in the stochastic signal model.

Statistical Bounds

Once a model for the signal has been chosen, it is possible to bound the accuracy of estimators of the parameters of the model. There are many techniques for determining statistical bounds; in this section we will derive bounds using two closely related methods: the Cramér-Rao approach [5], and a generalization of this approach due to Bhattacharyya [6].

The Cramér-Rao Bound

The simplest and most widely used of the statistical bounds on estimation is the Cramér-Rao bound. Using the notation $\langle x \rangle$ to denote the expected value of x , we may define a $K \times K$ matrix \mathbf{J} , derived from a set of K parameters $\{\Theta_{m_1}, \dots, \Theta_{m_K}\}$ of the model. The elements of \mathbf{J} are given by

$$J(i, j) \equiv \left\langle \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta_{m_i}} \cdot \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta_{m_j}} \right\rangle, \quad (2.5)$$

where any set of K model parameters may be chosen as long as the necessary derivatives exist and are linearly independent. The matrix \mathbf{J} is known as the Fisher information matrix. If a vector of partial derivatives \mathbf{v} is defined,

$$\mathbf{v} \equiv \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta} = \frac{1}{P(\Theta)} \left[\frac{\partial P(\Theta)}{\partial \Theta_{m_1}}, \frac{\partial P(\Theta)}{\partial \Theta_{m_2}}, \dots, \frac{\partial P(\Theta)}{\partial \Theta_{m_k}} \right]^T, \quad (2.6)$$

then \mathbf{J} can also be written in a more compact form as $\mathbf{J} = \langle \mathbf{v} \mathbf{v}^T \rangle$. Since the partial derivatives in \mathbf{v} are required to be linearly independent, \mathbf{J} is nonsingular. The inverse of the Fisher information matrix determines a lower limit for the variance of

any unbiased estimator; this limit, the Cramér-Rao bound, is given by the following theorem:

THEOREM 2.1 (CRAMÉR-RAO BOUND) *If $\hat{\Theta}_i$ is an unbiased estimator of a parameter Θ_i of a random process, $\mathbf{J} = \langle \mathbf{v}\mathbf{v}^T \rangle$ is a nonsingular $n \times n$ matrix constructed from the likelihood function $P(\Theta)$ of the random process according to equation (2.6), and all second partial derivatives of $P(\Theta)$ with respect to the included Θ_i are continuous, then $\text{var}(\hat{\Theta}_i) \geq \mathbf{J}^{-1}(i, i)$ for all $i \in \{1 \dots n\}$.*

(Since the Cramér-Rao bound is a special case of the Bhattacharyya bound, the proof will be deferred until the next section.)

The derivation of the Cramér-Rao bound does not require that all the parameters of $P(\Theta)$ be included in \mathbf{J} . It is frequently the case that only a subset of the parameters are of interest; for example, in frequency estimation, the problem is to estimate the ω_i . The remaining parameters, in this case the a_i and ϕ_i , and σ^2 , need not be included in \mathbf{J} to determine a bound for the frequency estimates. However, including all the parameters of the model yields a tighter bound, as shown by the following theorem, a slight generalization of a result due to Rife and Boorstyn [7].

THEOREM 2.2 *Let \mathbf{a} and \mathbf{b} be real random vectors of lengths n and $n+1$, respectively, with n of their elements identical, and let $\mathbf{\Pi}$ be a permutation matrix that rearranges \mathbf{b} so that the first n elements of $\mathbf{\Pi}\mathbf{b}$ are identical to \mathbf{a} :*

$$\begin{bmatrix} \mathbf{a} \\ \alpha \end{bmatrix} = \mathbf{\Pi}\mathbf{b}.$$

If $\mathbf{A} = \langle \mathbf{a}\mathbf{a}^T \rangle$, $\mathbf{B} = \langle \mathbf{b}\mathbf{b}^T \rangle$, both \mathbf{A}^{-1} and \mathbf{B}^{-1} exist, and $\mathbf{C} = \mathbf{\Pi}\mathbf{B}^{-1}\mathbf{\Pi}^T$, then $C(i, i) \geq A^{-1}(i, i)$ for all $i \in \{1 \dots n\}$.

PROOF: Since

$$\mathbf{\Pi B \Pi}^T = \begin{pmatrix} \mathbf{aa}^T & \alpha \mathbf{a} \\ \alpha \mathbf{a}^T & \alpha^2 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{v} \\ \mathbf{v}^T & \gamma \end{pmatrix},$$

using the partitioned matrix inversion lemma and the orthogonality of $\mathbf{\Pi}$, we find

$$\mathbf{\Pi B}^{-1} \mathbf{\Pi}^T = \begin{pmatrix} \mathbf{A}^{-1} + \beta \mathbf{A}^{-1} \mathbf{v} \mathbf{v}^T \mathbf{A}^{-1} & -\beta \mathbf{A}^{-1} \mathbf{v} \\ -\beta \mathbf{v}^T \mathbf{A}^{-1} & \beta \end{pmatrix},$$

where $\beta = (\gamma - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{v})^{-1}$. The leading $n \times n$ submatrix of \mathbf{C} is $\mathbf{A}^{-1} + \beta \mathbf{A}^{-1} \mathbf{v} \mathbf{v}^T \mathbf{A}^{-1}$, and for $C(i, i) \geq A^{-1}(i, i)$ for $i \in \{1 \dots n\}$ it is necessary and sufficient that the diagonal elements of $\beta \mathbf{D} = \beta \mathbf{A}^{-1} \mathbf{v} \mathbf{v}^T \mathbf{A}^{-1}$ are greater than or equal to zero. \mathbf{A} is symmetric, since $\langle a_i a_j \rangle = \langle a_j a_i \rangle$; its inverse is therefore symmetric, and $\mathbf{A}^{-1} \mathbf{v} \mathbf{v}^T \mathbf{A}^{-1} = (\mathbf{A}^{-1} \mathbf{v})(\mathbf{v}^T \mathbf{A}^{-T}) = \mathbf{c} \mathbf{c}^T = \mathbf{D}$. Because \mathbf{D} is an outer product, it is positive semidefinite, and therefore $\mathbf{D}(i, i) \geq 0$. In addition, $1/\beta = \gamma - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{v}$ is greater than or equal to zero, since $\langle (\alpha - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{a})^2 \rangle \geq 0$ and

$$\begin{aligned} \langle (\alpha - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{a})^2 \rangle &= \langle \alpha^2 - 2\mathbf{v}^T \mathbf{A}^{-1} \mathbf{a} \alpha + \mathbf{v}^T \mathbf{A}^{-1} \mathbf{a} \mathbf{a}^T \mathbf{A}^{-1} \mathbf{v} \rangle \\ &= \langle \alpha^2 \rangle - 2\mathbf{v}^T \mathbf{A}^{-1} \langle \alpha \mathbf{a} \rangle + \mathbf{v}^T \mathbf{A}^{-1} \langle \mathbf{a} \mathbf{a}^T \rangle \mathbf{A}^{-1} \mathbf{v} \\ &= \gamma - \mathbf{v}^T \mathbf{A}^{-1} \mathbf{v}. \end{aligned}$$

The fact that $\mathbf{\Pi B}^{-1} \mathbf{\Pi}^T$ must be invertible implies that $\beta > 0$. Therefore, because $C(i, i) = A^{-1}(i, i) + \beta D(i, i)$, and both β and $D(i, i)$ are greater than or equal to zero, $C(i, i) \geq A^{-1}(i, i)$ for all $i \in \{1 \dots n\}$. \square

Since the Cramér-Rao bound is derived from an outer product of the form treated above, this result is useful in a variety of situations; we will see in the following section that the Bhattacharyya bound also has this form. Two important corollaries

are given here. First, application of Theorem 2.2 and Theorem 2.1 indicates how the tightest possible Cramér-Rao bound may be obtained.

COROLLARY 2.3 *The tightest Cramér-Rao bound on the variance of an unbiased estimator $\hat{\Theta}_i$ of a parameter Θ_i of a random process is obtained when the derivatives of $P(\Theta)$ with respect to all parameters of the random process are included in \mathbf{v} , as long as the resulting \mathbf{J} is nonsingular.*

A similar result applies when there is more than one signal present, since additional signals simply mean that the model has more parameters.

COROLLARY 2.4 *The Cramér-Rao bound on the variance of an unbiased estimator of the frequency of a single sinusoid is also a lower bound for the variance of any unbiased estimator of the frequency of that sinusoid when other sinusoids are present.*

The proof of these corollaries is by induction using Theorem 2.2. The fact that a bound for one or two signals is also a bound for many signals is extremely important in practice, because it dramatically reduces the amount of computation required to produce a usable bound; rather than considering all the signals present, only those closest in frequency to the signal in question are used to compute the bound. When two closely spaced sinusoids are well separated from the other signals, this approach yields a good approximation to the exact bound.

The Bhattacharyya Bound

The Cramér-Rao bound may be viewed as the simplest form of a more general bound, first developed by Bhattacharyya [6]. Define a generalization of the derivative vector \mathbf{v} that includes derivatives of higher orders:

$$\mathbf{w} \equiv \frac{1}{P(\Theta)} \left[\frac{\partial P(\Theta)}{\partial \Theta_i}, \dots, \frac{\partial^2 P(\Theta)}{\partial \Theta_{j_1} \partial \Theta_{j_2}}, \dots, \frac{\partial^m P(\Theta)}{\partial \Theta_{k_1} \partial \Theta_{k_2} \dots \partial \Theta_{k_m}}, \dots \right]^T. \quad (2.7)$$

As before, all of the partial derivatives included in \mathbf{w} must exist and be linearly independent; to simplify the notation, we will also assume that all the first partial derivatives of $P(\Theta)$ are grouped together at the head of \mathbf{w} , as indicated in equation (2.7). When $P(\Theta)$ has continuous n th derivatives with respect to the parameters in Θ ,

$$\frac{\partial^n P(\Theta)}{\partial \Theta_{i_1} \dots \partial \Theta_{i_n}} = \frac{\partial^n P(\Theta)}{\partial \Theta_{k_1} \dots \partial \Theta_{k_n}},$$

where $\{k_1 \dots k_n\}$ is any permutation of $\{i_1 \dots i_n\}$. This means that only one derivative of a given order with respect to a given set of parameters may be included in \mathbf{w} . As in the case of the Cramér-Rao bound, the tightest bound for a given maximum order is obtained when all linearly independent derivatives of this order or less are included. If the highest order included in \mathbf{w} is m , then the matrix \mathbf{K}_m is defined as

$$\mathbf{K}_m \equiv \langle \mathbf{w} \mathbf{w}^T \rangle. \quad (2.8)$$

By analogy with the Fisher information matrix, \mathbf{K}_m is referred to as the m th order Bhattacharyya information matrix; the first order Bhattacharyya information matrix is just the Fisher information matrix.

THEOREM 2.5 (BHATTACHARYYA BOUND) *If $\hat{\Theta}_i$ is an unbiased estimator of a parameter Θ_i of a random process, $\mathbf{K} = \langle \mathbf{w} \mathbf{w}^T \rangle$ is a nonsingular $n \times n$ matrix constructed from the likelihood function $P(\Theta)$ of the random process according to equation (2.7), where the first k elements of \mathbf{w} contain all of the included first partial derivatives, the highest order partial derivative contained in \mathbf{w} is m , and all partial derivatives of order $m + 1$ of $P(\Theta)$ with respect to the Θ_i are continuous, then $\text{var}(\hat{\Theta}_i) \geq \mathbf{K}^{-1}(i, i)$ for all $i \in \{1 \dots n\}$.*

PROOF: Consider the matrix given by

$$\hat{\mathbf{A}} = \left\langle \begin{bmatrix} (\hat{\Theta} - \Theta) \\ \mathbf{w} \end{bmatrix} \cdot \begin{bmatrix} (\hat{\Theta} - \Theta)^T & \mathbf{w}^T \end{bmatrix} \right\rangle.$$

where Θ and $\hat{\Theta}$ contain the same parameters in the same order, and the order is chosen so that it matches the order of the first partial derivatives in \mathbf{w} , so that if

$$\mathbf{w} = \frac{1}{P(\Theta)} \left[\frac{\partial P(\Theta)}{\partial \Theta_1}, \frac{\partial P(\Theta)}{\partial \Theta_2}, \dots, \frac{\partial P(\Theta)}{\partial \Theta_k}, \dots \right],$$

then $\Theta = [\Theta_1, \Theta_2, \dots, \Theta_k]^T$, and $\hat{\Theta} = [\hat{\Theta}_1, \hat{\Theta}_2, \dots, \hat{\Theta}_k]^T$. Since $\hat{\Theta}$ is an unbiased estimator, $\langle (\hat{\Theta}_i - \Theta_i) \rangle = 0$. Taking partial derivatives,

$$\frac{\partial^m}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} \langle (\hat{\Theta}_i - \Theta_i) \rangle = 0.$$

If $P(\Theta)$ has $m + 1$ continuous partial derivatives, the order of differentiation and expectation (that is, integration) may be interchanged, giving

$$\begin{aligned} \frac{\partial^m}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} \langle (\hat{\Theta}_i - \Theta_i) \rangle &= \int \frac{\partial^m ((\hat{\Theta}_i - \Theta_i)p(\tilde{\mathbf{y}}|\Theta, \mathbf{y}))}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} d\mathbf{y} \\ &= - \int \frac{\partial^m \Theta_i}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} p(\tilde{\mathbf{y}}|\Theta, \mathbf{y}) d\mathbf{y} \\ &\quad - \int \frac{\partial^m p(\tilde{\mathbf{y}}|\Theta, \mathbf{y})}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} \Theta_i d\mathbf{y} \\ &= - \int \delta_{ij} p(\tilde{\mathbf{y}}|\Theta, \mathbf{y}) d\mathbf{y} \\ &\quad - \int \frac{1}{P(\Theta)} \frac{\partial^m P(\Theta)}{\partial \Theta_{i_1} \partial \Theta_{i_2} \dots \partial \Theta_{i_m}} \Theta_i P(\Theta) d\mathbf{y} \\ &= -\delta_{ij} - \langle w_j \cdot \Theta_i \rangle \\ &= 0, \end{aligned}$$

where we have used the index j to indicate the element of \mathbf{w} associated with the given partial derivative. This means that $-\delta_{ij} = \langle w_j \Theta_i \rangle$, or

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{C}_{\hat{\Theta}} & -\mathbf{Z} \\ -\mathbf{Z} & \mathbf{K} \end{bmatrix},$$

where

$$\mathbf{Z} = \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \end{bmatrix}.$$

Because it is an autocorrelation matrix, $\tilde{\mathbf{A}}$ is symmetric and positive semidefinite; it can be written as

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_k & -\mathbf{Z}\mathbf{K}^{-1} \\ \mathbf{0} & \mathbf{I}_{n-k} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{\hat{\Theta}} - \mathbf{Z}\mathbf{K}^{-1}\mathbf{Z} & \mathbf{0} \\ \mathbf{0} & \mathbf{K} \end{bmatrix} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ -\mathbf{K}^{-1}\mathbf{Z}^T & \mathbf{I}_{n-k} \end{bmatrix}.$$

By the Sylvester inertia principle, $\mathbf{C}_{\hat{\Theta}} - \mathbf{Z}\mathbf{K}^{-1}\mathbf{Z}$ is also positive semidefinite, and so its diagonal elements are greater than or equal to zero. Since the diagonal elements of $\mathbf{C}_{\hat{\Theta}}$ are the variances of the estimates, and $\mathbf{Z}\mathbf{K}^{-1}\mathbf{Z}$ is the $k \times k$ leading principal submatrix of \mathbf{K}^{-1} , $\text{var}(\hat{\Theta}_i) - K^{-1}(i, i) \geq 0$. \square

From Theorem 2.2, we can see that the tightest Bhattacharyya bound of a given maximum order is obtained by including all the linearly independent partial derivatives of that order or less. Two additional important corollaries are:

COROLLARY 2.6 *The Bhattacharyya bound on the variance of an unbiased estimator of the frequency of a single sinusoid is also a lower bound for the variance of any unbiased estimator of the frequency of that sinusoid when other sinusoids are present.*

COROLLARY 2.7 *The Bhattacharyya bound on the variance of an unbiased estimator is at least as tight as the Cramér-Rao bound obtained using the same set of first derivatives.*

We will now describe the computation of these bounds for the sinusoidal signal model.

Derivation of Bounds

The calculation of the bounds described above for a given signal is not conceptually difficult, but the algebraic manipulations required when using the deterministic signal model can become extremely complex. The computation of the Cramér-Rao and second order Bhattacharyya bounds for frequency estimators of signals with one and two real sinusoidal components in additive white Gaussian noise is discussed in the following sections.

It is often convenient in the derivation of bounds to employ the logarithm of the likelihood function, $F(\Theta) \equiv \log(P(\Theta))$, referred to as the log-likelihood function. For real sinusoidal signals in white Gaussian noise, $F(\Theta)$ is given by

$$F(\Theta) = -\frac{L}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{k=k_0+1}^{k_0+L} \left(y[k] - \sum_{i=1}^N a_i \cos(\omega_i k + \phi_i) \right)^2. \quad (2.9)$$

The utility of the log-likelihood function arises from the identities

$$\begin{aligned} \frac{\partial F(\Theta)}{\partial \Theta_i} &= \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta_i}, \\ \frac{\partial^2 F(\Theta)}{\partial \Theta_i \partial \Theta_j} + \frac{\partial F(\Theta)}{\partial \Theta_i} & \\ \frac{\partial F(\Theta)}{\partial \Theta_j} &= \frac{1}{P(\Theta)} \frac{\partial^2 P(\Theta)}{\partial \Theta_i \partial \Theta_j} \end{aligned}$$

used in the derivation of the vectors \mathbf{v} and \mathbf{w} , which determine the Cramér-Rao and Bhattacharyya bounds.

Derivation of the Cramér-Rao Bound

Cramér-Rao bounds for the estimation of the parameters of sinusoids were first derived by Rife and Boorstyn [1, 7]. Several useful asymptotic approximations for the Cramér-Rao bound have been derived by Stoica and Nehorai [2, 8]. The Cramér-Rao bound is determined by the inverse of the Fisher information matrix \mathbf{J} , whose elements are $J(i, j) \equiv \langle v_i v_j \rangle$. In terms of the log-likelihood function,

$$\langle v_i v_j \rangle = \left\langle \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta_i} \cdot \frac{1}{P(\Theta)} \frac{\partial P(\Theta)}{\partial \Theta_j} \right\rangle = \left\langle \frac{\partial F(\Theta)}{\partial \Theta_i} \cdot \frac{\partial F(\Theta)}{\partial \Theta_j} \right\rangle.$$

For the tightest possible bound, all of the unknown parameters of the model signal \mathbf{y} must be included in \mathbf{v} :

$$\mathbf{v} = \left[\frac{\partial F}{\partial \sigma^2}, \frac{\partial F}{\partial a_1}, \frac{\partial F}{\partial \omega_1}, \frac{\partial F}{\partial \phi_1}, \dots, \frac{\partial F}{\partial a_N}, \frac{\partial F}{\partial \omega_N}, \frac{\partial F}{\partial \phi_N} \right]^T.$$

As long as all of the sinusoidal frequencies are different, \mathbf{J} will be nonsingular, and since $\langle v_i v_j \rangle = \langle v_j v_i \rangle$, \mathbf{J} is symmetric.

To clarify which elements of \mathbf{v} and \mathbf{J} are being referred to, a condensed notation will be employed, where $v_{(\Theta_i)}$ will stand for $\frac{\partial F}{\partial \Theta_i}$, $J_{(\Theta_i; \Theta_j)}$ will stand for $\langle v_{(\Theta_i)} v_{(\Theta_j)} \rangle$, and the parameters of the model will be shown explicitly as amplitudes, frequencies, phases, or standard deviations. For example:

$$J_{(a_i; \omega_j)} \equiv \langle v_{(a_i)} v_{(\omega_j)} \rangle = \left\langle \frac{\partial F}{\partial a_i} \cdot \frac{\partial F}{\partial \omega_j} \right\rangle.$$

The elements of \mathbf{v} are given by the following partial derivatives:

$$\begin{aligned} v_{(\sigma^2)} &= \frac{1}{2\sigma^2} \left(\frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k]^2 - L \right) \\ v_{(a_i)} &= \frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \cos(\omega_i k + \phi_i) \end{aligned}$$

$$\begin{aligned}
 v_{(\omega_i)} &= -\frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] k \sin(\omega_i k + \phi_i) \\
 v_{(\phi_i)} &= -\frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \sin(\omega_i k + \phi_i)
 \end{aligned}$$

In each of the expectations $\langle v_i v_j \rangle$ which determine \mathbf{J} , the only random quantities are the $n[k]$, which are independent zero mean Gaussian random variables. To simplify the calculation of expected values of the products of the derivatives, the following results are useful:

$$\begin{aligned}
 \langle n[k] \rangle &= 0 \\
 \langle n[k] n[l] \rangle &= \begin{cases} \sigma^2, & k = l \\ 0, & k \neq l \end{cases} \\
 \langle n[k]^2 n[l] \rangle &= 0 \\
 \langle n[k]^2 n[l] n[m] \rangle &= \begin{cases} 3\sigma^4, & k = l = m \\ \sigma^4, & k \neq l = m \\ 0, & l \neq m \end{cases}
 \end{aligned}$$

Because \mathbf{J} is symmetric, it is only necessary to calculate the elements in its upper triangle. These are given by:

$$\begin{aligned}
 J_{(\sigma^2; \sigma^2)} &= \frac{L}{2\sigma^4} \\
 J_{(\sigma^2; a_i)} &= 0 \\
 J_{(\sigma^2; \omega_i)} &= 0 \\
 J_{(\sigma^2; \phi_i)} &= 0 \\
 J_{(a_i; a_j)} &= \frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \cos(\omega_i k + \phi_i) \cos(\omega_j k + \phi_j) \\
 J_{(a_i; \omega_j)} &= -\frac{a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k \cos(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j)
 \end{aligned}$$

$$\begin{aligned}
J_{(a_i; \phi_j)} &= -\frac{a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \cos(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
J_{(\omega_i; \omega_j)} &= \frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k^2 \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
J_{(\omega_i; \phi_j)} &= \frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
J_{(\phi_i; \phi_j)} &= \frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j)
\end{aligned}$$

Since the expected value of the product of $v_{(\sigma^2)}$ and any other element of \mathbf{v} is zero, \mathbf{J} is block diagonal. Using the fact that

$$\begin{bmatrix} \alpha & 0 \\ 0 & \mathbf{A} \end{bmatrix}^{-1} = \begin{bmatrix} 1/\alpha & 0 \\ 0 & \mathbf{A}^{-1} \end{bmatrix},$$

the Cramér-Rao bound on estimation of σ^2 may be found immediately to be $\text{var}(\hat{\sigma}^2) \geq 2\sigma^4/L$. The inverse of the submatrix of \mathbf{J} containing the other parameters then determines the bounds on estimation of the magnitudes, frequencies, and phases.

It is possible to simplify the sums of trigonometric expressions in the elements of \mathbf{J} to remove the summations over k , but the resulting expressions are still complex, and little insight is gained by the simplification. (The results are given by Porat [4, pp. 262–264].) For signals with more than one or two sinusoidal components, the most effective approach is to calculate \mathbf{J} using the expressions above, and numerically compute its inverse to find the Cramér-Rao bounds.

For the case of a single sinusoid, however, useful expressions may be analytically derived. In this case, the bounds are determined by a submatrix of \mathbf{J} :

$$\frac{1}{\sigma^2} \begin{bmatrix} \sum \cos^2(\omega k + \phi) & -a/2 \sum k \sin(2\omega k + 2\phi) & -a/2 \sum \sin(2\omega k + 2\phi) \\ -a/2 \sum k \sin(2\omega k + 2\phi) & a^2 \sum k^2 \sin^2(\omega k + \phi) & a^2 \sum k \sin^2(\omega k + \phi) \\ -a/2 \sum \sin(2\omega k + 2\phi) & a^2 \sum k \sin^2(\omega k + \phi) & a^2 \sum \sin^2(\omega k + \phi) \end{bmatrix}$$

The Cramér-Rao bounds on estimation of a , ω , and ϕ are given by the diagonal elements of the inverse of this matrix; by setting terms that are small for large L to zero and computing the inverse, an asymptotic Cramér-Rao bound, useful for large data records, is obtained.

A surprising feature of the result of this computation, first noted by Rife [1], is that the bound on the variance of the phase estimate depends on the location of the point $k = 0$ in relation to the sample (i.e., the value of k_0). If the first sample is chosen to be $k = 0$, then the off-diagonal terms given by the sum $a^2 \sum k \sin^2(\omega k + \phi)$ do not become negligible as L becomes large; the value of the diagonal term $J_{(\omega, \omega)}$ is also affected by the choice of origin. The fact that the off-diagonal terms do not become negligible for large L corresponds to a coupling between the frequency and phase estimates. If the origin is chosen to be at the center of the observed data, the coupling term approaches zero as L increases, indicating that the frequency and phase estimates are asymptotically decoupled.

Although this effect is surprising, it has a natural explanation. While the frequency and amplitude of a sinusoid are translation invariant, the phase is defined with respect to a fixed location, the point $k = 0$. The mechanism responsible for the coupling between phase and frequency becomes clear if we consider the problem of estimating the phase of a signal from data on an interval that does not include $k = 0$. It is more difficult to estimate the phase of a signal at a point far from the center of the measurement interval because this requires some degree of extrapolation, which increases the uncertainty of the phase estimate. The coupling between the phase and frequency estimates is simply a consequence of the fact that knowledge of the frequency is necessary to determine the phase.

The asymptotic value of the Cramér-Rao bound for the frequency estimate is not altered by the choice of origin, but the bound for phase estimation is affected; in accordance with the intuitive justification given above, it can be shown that the

minimum value is obtained when the origin is at the center of the data record [1]. The asymptotic bound for phase estimation is increased by a factor of four if $k = 0$ is taken to be at the start of the data record, rather than at the center; this effect has apparently been neglected in several derivations of the Cramér-Rao bound for phase estimation [3, 9].

For finite L , the bounds for both frequency and phase depend on the choice of origin, and the minimum value is reached when the origin is taken to be the center of the measurement interval. Under this assumption, as the number of samples L becomes large, the matrix can be approximated by

$$\frac{1}{\sigma^2} \begin{bmatrix} L/2 & 0 & 0 \\ 0 & a^2 L(L-1)^2/12 & 0 \\ 0 & 0 & a^2 L/2 \end{bmatrix},$$

and the Cramér-Rao bounds on estimation of a , ω , and ϕ approach the diagonal elements of

$$\sigma^2 \begin{bmatrix} 2/L & 0 & 0 \\ 0 & 24/a^2 L(L-1)^2 & 0 \\ 0 & 0 & 2L/a^2 \end{bmatrix}.$$

The asymptotic (for large L) Cramér-Rao bounds on the estimation of the parameters of a single real sinusoid in additive white Gaussian noise are therefore given by

$$\begin{aligned} \text{var}(\hat{a}) &\geq \frac{2\sigma^2}{L}, \\ \text{var}(\hat{\omega}) &\geq \frac{24\sigma^2}{a^2 L^3}, \\ \text{var}(\hat{\phi}) &\geq \frac{2\sigma^2}{a^2 L}. \end{aligned}$$

In terms of the signal-to-noise ratio $\eta = a^2/2\sigma^2$, the bound on the accuracy of the frequency estimate is

$$\text{var}(\hat{\omega}) \geq \frac{12}{\eta L^3},$$

or, in terms of the cycle frequency f ,

$$\text{var}(\hat{f}) \geq \frac{3}{\pi^2 \eta L^3}.$$

Note that the minimum variance of any unbiased frequency estimate decreases asymptotically as the cube of the number of samples. This is a powerful argument for maximizing the number of samples used, since it indicates that the simultaneous use of many consecutive samples allows the variance to be reduced very rapidly. If the samples are not consecutive, as in the multiple snapshot case in bearing estimation, the variance decreases only as the inverse of the number of snapshots [10].

Derivation of the Bhattacharyya Bound

To calculate the Bhattacharyya bound, additional higher-order derivatives of the likelihood function are needed. Since $P(\Theta)$ is infinitely differentiable, all partial derivatives with respect to the same set of parameters are equal. For second derivatives, this means that, for example, $\frac{\partial^2 P(\Theta)}{\partial a_i \partial \omega_j} = \frac{\partial^2 P(\Theta)}{\partial \omega_j \partial a_i}$. For the Bhattacharyya information matrix \mathbf{K} to be nonsingular, only one derivative with respect to each set of parameters may be included.

The elements of \mathbf{w} and \mathbf{K} will be referenced with the same notation used for \mathbf{v} and \mathbf{J} , so that

$$w_{(\Theta_i, \Theta_j)} \equiv \frac{1}{P(\Theta)} \frac{\partial^2 P(\Theta)}{\partial \Theta_i \partial \Theta_j},$$

and

$$K_{(\Theta_i, \Theta_j, \Theta_k, \Theta_l)} \equiv \langle w_{(\Theta_i, \Theta_j)} w_{(\Theta_k, \Theta_l)} \rangle.$$

(Note that $w_{(\Theta_i)} = v_{(\Theta_i)}$ and $K_{(\Theta_i, \Theta_j)} = J_{(\Theta_i, \Theta_j)}.$)

In terms of the log-likelihood function, \mathbf{w} is given by

$$w_{(\Theta_i, \Theta_j)} = \frac{1}{P(\Theta)} \frac{\partial^2 P(\Theta)}{\partial \Theta_i \partial \Theta_j} = \frac{\partial^2 F(\Theta)}{\partial \Theta_i \partial \Theta_j} + \frac{\partial F(\Theta)}{\partial \Theta_i} \frac{\partial F(\Theta)}{\partial \Theta_j}.$$

For the tightest possible second order Bhattacharyya bound, all linearly independent first and second derivatives of $P(\Theta)$ must be included in \mathbf{w} . These are given by:

$$\begin{aligned} w_{(\sigma^2)} &= \frac{1}{2\sigma^2} \left(\frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k]^2 - L \right) \\ w_{(a_i)} &= \frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \cos(\omega_i k + \phi_i) \\ w_{(\omega_i)} &= -\frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] k \sin(\omega_i k + \phi_i) \\ w_{(\phi_i)} &= -\frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \sin(\omega_i k + \phi_i) \\ w_{(\sigma^2, \sigma^2)} &= \frac{L^2 + 2L}{4\sigma^4} - \frac{L + 2}{2\sigma^6} \sum_{k=k_0+1}^{k_0+L} n[k]^2 + \frac{1}{4\sigma^8} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l]^2 \\ w_{(\sigma^2, a_i)} &= -\frac{L + 2}{2\sigma^4} \sum_{k=k_0+1}^{k_0+L} n[k] \cos(\omega_i k + \phi_i) \\ &\quad + \frac{1}{2\sigma^6} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l] \cos(\omega_i l + \phi_i) \\ w_{(\sigma^2, \omega_i)} &= a_i \frac{L + 2}{2\sigma^4} \sum_{k=k_0+1}^{k_0+L} n[k] k \sin(\omega_i k + \phi_i) \\ &\quad - \frac{a_i}{2\sigma^6} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l] l \sin(\omega_i l + \phi_i) \\ w_{(\sigma^2, \phi_i)} &= a_i \frac{L + 2}{2\sigma^4} \sum_{k=k_0+1}^{k_0+L} n[k] \sin(\omega_i k + \phi_i) \end{aligned}$$

$$\begin{aligned}
& -\frac{a_i}{2\sigma^6} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l] \sin(\omega_i k + \phi_i) \\
w_{(a_i, a_j)} &= -\frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \cos(\omega_i k + \phi_i) \cos(\omega_j k + \phi_j) \\
& + \frac{1}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] \cos(\omega_i k + \phi_i) \cos(\omega_j l + \phi_j) \\
w_{(a_i, \omega_j)} &= \frac{a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k \cos(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
& - \delta_{ij} \frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] k \sin(\omega_i k + \phi_i) \\
& - \frac{a_j}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] l \cos(\omega_i k + \phi_i) \sin(\omega_j l + \phi_j) \\
w_{(a_i, \phi_j)} &= \frac{a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \cos(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
& - \delta_{ij} \frac{1}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \sin(\omega_i k + \phi_i) \\
& - \frac{a_j}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] \cos(\omega_i k + \phi_i) \sin(\omega_j l + \phi_j) \\
w_{(\omega_i, \omega_j)} &= -\frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k^2 \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
& - \delta_{ij} \frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] k^2 \cos(\omega_i k + \phi_i) \\
& + \frac{a_i a_j}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] k l \sin(\omega_i k + \phi_i) \sin(\omega_j l + \phi_j) \\
w_{(\omega_i, \phi_j)} &= -\frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} k \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j) \\
& - \delta_{ij} \frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] k \cos(\omega_i k + \phi_i) \\
& + \frac{a_i a_j}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] k \sin(\omega_i k + \phi_i) \sin(\omega_j l + \phi_j) \\
w_{(\phi_i, \phi_j)} &= -\frac{a_i a_j}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} \sin(\omega_i k + \phi_i) \sin(\omega_j k + \phi_j)
\end{aligned}$$

$$\begin{aligned}
& - \delta_{ij} \frac{a_i}{\sigma^2} \sum_{k=k_0+1}^{k_0+L} n[k] \cos(\omega_i k + \phi_i) \\
& + \frac{a_i a_j}{\sigma^4} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] \sin(\omega_i k + \phi_i) \sin(\omega_j l + \phi_j)
\end{aligned}$$

To simplify the calculation of \mathbf{K} from \mathbf{w} , it is helpful to place the elements of \mathbf{w} in a standard form. Each element of \mathbf{w} may be written as:

$$\begin{aligned}
w_{(\Theta_i, \Theta_j)} &= A_{(\Theta_i, \Theta_j)} \\
&+ \sum_{k=k_0+1}^{k_0+L} n[k] B_{(\Theta_i, \Theta_j)}[k] \\
&+ \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k] n[l] C_{(\Theta_i, \Theta_j)}[k, l] \\
&+ D_{(\Theta_i, \Theta_j)} \sum_{k=k_0+1}^{k_0+L} n[k]^2 \\
&+ \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l] E_{(\Theta_i, \Theta_j)}[l] \\
&+ F_{(\Theta_i, \Theta_j)} \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} n[k]^2 n[l]^2
\end{aligned}$$

Using this form, each element of \mathbf{K} may be calculated from the coefficients of the two relevant elements of \mathbf{w} . The value of $K_{(\Theta_i, \Theta_j, \Theta_k, \Theta_l)}$ is the sum of the expected values of the components of the cross product of $w_{(\Theta_i, \Theta_j)}$ and $w_{(\Theta_k, \Theta_l)}$. Since each w is a sum of terms with the coefficients A, B, C, D, E , and F defined above, the cross product can be expressed in terms of these coefficients. The expected value of the product of the A term in $w_{(\Theta_i, \Theta_j)}$ and the C term in $w_{(\Theta_k, \Theta_l)}$ will be denoted K_{AC} , and similarly for each of the other terms. Using the fact that the expected value of any odd power of $n[k]$ is zero, we can easily see that many of these components are zero, as shown in the chart below.

	$A_{(\Theta_i, \Theta_j)}$	$B_{(\Theta_i, \Theta_j)}$	$C_{(\Theta_i, \Theta_j)}$	$D_{(\Theta_i, \Theta_j)}$	$E_{(\Theta_i, \Theta_j)}$	$F_{(\Theta_i, \Theta_j)}$
$A_{(\Theta_k, \Theta_l)}$	K_{AA}	0	K_{AC}	K_{AD}	0	K_{AF}
$B_{(\Theta_k, \Theta_l)}$	0	K_{BB}	0	0	K_{BE}	0
$C_{(\Theta_k, \Theta_l)}$	K_{CA}	0	K_{CC}	K_{CD}	0	K_{CF}
$D_{(\Theta_k, \Theta_l)}$	K_{DA}	0	K_{DC}	K_{DD}	0	K_{DF}
$E_{(\Theta_k, \Theta_l)}$	0	K_{EB}	0	0	K_{EE}	0
$F_{(\Theta_k, \Theta_l)}$	K_{FA}	0	K_{FC}	K_{FD}	0	K_{FF}

To determine the values of each of these components, we must calculate expected values of summations of products of the $n[k]$. In any quadruple summation with indices ranging over L values, there will be

- L terms with all 4 indices identical,
- $4L(L-1)$ terms with only 3 indices identical,
- $3L(L-1)$ terms with 2 pairs of indices identical,
- $6L(L-1)(L-2)$ terms with only 1 pair of identical indices, and
- $L(L-1)(L-2)(L-3)$ terms with all indices different.

Similarly, in any triple summation, there will be

- L terms with all 3 indices identical,
- $3L(L-1)$ terms with only 2 indices identical, and
- $L(L-1)(L-2)$ terms with all indices different.

Finally, in addition to the expected values of products of the $n[k]$ listed earlier, we will also require the following values:

$$\begin{aligned}
 \langle n[i]^2 n[j]^2 n[k] \rangle &= 0 \\
 \langle n[i]^2 n[j]^2 n[k]^2 \rangle &= \begin{cases} 15\sigma^6, & i = j = k \\ 3\sigma^6, & \text{any two indices equal} \\ \sigma^6, & i \neq j \neq k \end{cases} \\
 \langle n[i]^2 n[j]^2 n[k]^2 n[l] \rangle &= 0 \\
 \langle n[i]^2 n[j]^2 n[k]^2 n[l]^2 \rangle &= \begin{cases} 105\sigma^8, & i = j = k = l \\ 15\sigma^8, & \text{any three indices equal} \\ 9\sigma^8, & \text{two pair of indices equal} \\ 3\sigma^8, & \text{one pair of indices equal} \\ \sigma^8, & i \neq j \neq k \neq l \end{cases}
 \end{aligned}$$

Using these properties, we can determine the value of each of the constants in the cross product. Since K_{XY} and K_{YX} are simply related, we will only list the unique elements:

$$\begin{aligned}
 K_{AA} &= A_{(\Theta_i, \Theta_j)} A_{(\Theta_k, \Theta_l)} \\
 K_{AC} &= \sigma^2 A_{(\Theta_k, \Theta_l)} \sum_{k=k_0+1}^{k_0+L} C_{(\Theta_i, \Theta_j)}[k, k] \\
 K_{AD} &= L\sigma^2 A_{(\Theta_k, \Theta_l)} D_{(\Theta_i, \Theta_j)} \\
 K_{AF} &= (L^2 + 2L) \sigma^4 A_{(\Theta_k, \Theta_l)} F_{(\Theta_i, \Theta_j)} \\
 K_{BB} &= \sigma^2 \sum_{k=k_0+1}^{k_0+L} B_{(\Theta_i, \Theta_j)}[k] B_{(\Theta_k, \Theta_l)}[k] \\
 K_{BE} &= (L + 2) \sigma^4 \sum_{k=k_0+1}^{k_0+L} B_{(\Theta_k, \Theta_l)}[k] E_{(\Theta_i, \Theta_j)}[k]
 \end{aligned}$$

$$\begin{aligned}
K_{CC} &= \sigma^4 \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} C_{(\Theta_i, \Theta_j)}[k, k] C_{(\Theta_k, \Theta_l)}[l, l] \\
&\quad + \sigma^4 \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} C_{(\Theta_i, \Theta_j)}[k, l] C_{(\Theta_k, \Theta_l)}[k, l] \\
&\quad + \sigma^4 \sum_{k=k_0+1}^{k_0+L} \sum_{l=k_0+1}^{k_0+L} C_{(\Theta_i, \Theta_j)}[k, l] C_{(\Theta_k, \Theta_l)}[l, k] \\
K_{CD} &= (L+2) \sigma^4 D_{(\Theta_i, \Theta_j)} \sum_{k=k_0+1}^{k_0+L} C_{(\Theta_k, \Theta_l)}[k, k] \\
K_{CF} &= (L^2 + 6L + 8) \sigma^6 F_{(\Theta_i, \Theta_j)} \sum_{k=k_0+1}^{k_0+L} C_{(\Theta_k, \Theta_l)}[k, k] \\
K_{DD} &= (L^2 + 2L) \sigma^4 D_{(\Theta_i, \Theta_j)} D_{(\Theta_k, \Theta_l)} \\
K_{DF} &= L(L^2 + 6L + 8) \sigma^6 D_{(\Theta_k, \Theta_l)} F_{(\Theta_i, \Theta_j)} \\
K_{EE} &= (L^2 + 6L + 8) \sigma^6 \sum_{k=k_0+1}^{k_0+L} E_{(\Theta_i, \Theta_j)}[k] E_{(\Theta_k, \Theta_l)}[k] \\
K_{FF} &= (L^4 + 12L^3 + 44L^2 + 48L) \sigma^8 F_{(\Theta_i, \Theta_j)} F_{(\Theta_k, \Theta_l)}
\end{aligned}$$

We now have the information necessary to calculate the Bhattacharyya bound for estimation of the amplitudes, frequencies, and phases of sinusoidal signals in white Gaussian noise. It is clearly impractical to calculate the bound analytically, but the equations above may be easily implemented using a computer to numerically calculate bounds for specific signals. It is important to note that, when the signal-to-noise ratio is high, or the number of samples very large, the Bhattacharyya information matrix may be ill conditioned; however, in this case the Bhattacharyya bound is very close to the Cramér-Rao bound, and numerical errors in the computation of the bound are easily detected. Nevertheless, it is wise to compute the bound in double precision, and monitor the condition of the Bhattacharyya matrix to ensure that the results are of acceptable accuracy.

The results of this calculation, and a discussion of the circumstances in which the Bhattacharyya bound is a significant improvement over the Cramér-Rao bound, are given in the following section.

Comparison of Bounds

We have seen above that the Bhattacharyya bounds on frequency estimation are tighter than the Cramér-Rao bounds; in this section, we will compare the two bounds to determine when the difference is significant. Since existing frequency estimation techniques approach the Cramér-Rao bound very closely when there are many samples, the signal-to-noise ratio is high, and the sinusoids in the input data are widely separated, we do not expect to find a significant difference between the two bounds under these circumstances. Substantial differences are found, however, in cases where the number of samples is small, the signal-to-noise ratio is low, or the data contain sinusoids whose frequencies are close.

First, we will examine the effect of variations in the number of samples on the Cramér-Rao and Bhattacharyya bounds. Figure 2.1 shows the Cramér-Rao bound and the Bhattacharyya bound on the variance of a frequency estimator for a single sinusoid as the number of samples varies. The parameters of the sinusoid are $a = 1$, $\omega = 0.2$, $\phi = 0$, and $\sigma^2 = 30$.

Figure 2.1 clearly shows the extremely rapid decrease in both bounds as the number of samples increases. This decrease is asymptotically proportional to L^3 , and the rate is close to this asymptotic limit even for relatively small data records. Variations about the asymptotic rate occur for small record lengths because of the effect of fractional cycles of the signal in the data record. Notice that the period of the perturbation around the asymptotic line is the same as the signal period, approximately 31 samples; in the example shown, the signal is $\cos(0.2k)$, and the bounds decrease most slowly when L is a multiple of 10π . The exact shape of the perturbation about the asymptotic limit is dependent on the frequency and the phase of the signal, but a similar effect occurs for any choice of these parameters.

The largest difference between the the Cramér-Rao and Bhattacharyya bounds occurs for small data records. For the example shown, the Bhattacharyya bound on

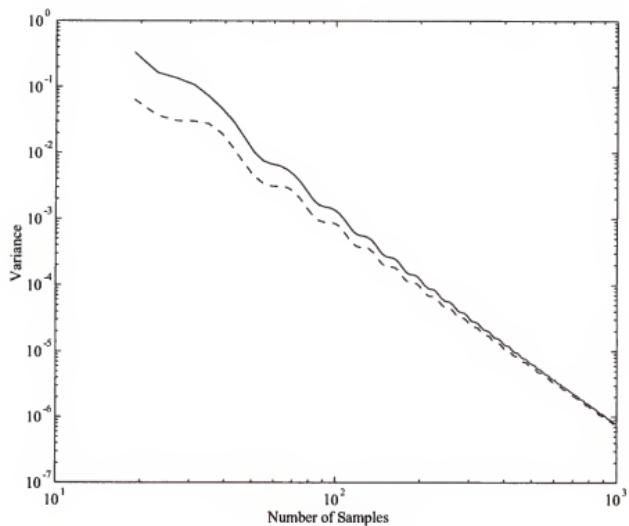


Figure 2.1: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds on the Variance of $\hat{\omega}$ for a Single Sinusoid

the variance is twice as large as the Cramér-Rao bound when $L = 60$, and five times as large when $L = 20$. The difference would of course be greater than this for smaller L , but bounds on the variance that are greater than 0.8 have little meaning; a variance of $\pi^2/12 \approx 0.822$ can be achieved with no computation at all, by randomly choosing a $\hat{\omega}$ uniformly distributed between 0 and π . As the number of samples becomes large, the two bounds approach equality; however, for finite L the Bhattacharyya bound is always larger than the Cramér-Rao bound. This substantiates the earlier assertion that no estimator of sinusoidal frequencies can be efficient for finite L .

Next, we will examine the effect of frequency variation on the bounds. Figure 2.2 shows the two bounds for a single signal with varying frequency, and $a = 1$, $\phi = 0$, $\sigma^2 = 0.1$, and $L = 19$. The two bounds differ significantly only for values of ω close to 0 or π . A detailed view of the low frequency region is shown in Figure 2.3.

The increase in the bounds on estimation when ω approaches 0 or π is a consequence of the fact that signals at these frequencies are sums of closely spaced *complex* sinusoids. The presence of another signal at a nearby frequency decreases the accuracy to which either frequency may be determined. We shall see below that a similar effect occurs for closely spaced real sinusoids.

The final example with a single sinusoid, Figure 2.4, shows the effect of changes in the signal-to-noise ratio. For this example, $a = 1$, $\omega = 0.2$, $\phi = 0$, and $L = 39$. Since the power in a real sinusoid with amplitude a is $a^2/2$, the signal-to-noise ratio is $a^2/2\sigma^2$.

The bounds on estimation of the frequency increase if additional sinusoids are added to the signal, and the change in the bounds depends on the frequency separation of the two sinusoids. Figure 2.5 shows the bounds for a sinusoid with the same parameters as used for Figure 2.4 ($a_1 = 1$, $\omega_1 = 0.2$, $\phi_1 = 0$), when a second sinusoid with parameters $a_2 = 1$, $\omega_2 = 0.3$, and $\phi_2 = 0$ is present, for $L = 39$. The addition of the second signal has caused the bounds on estimating the frequency of

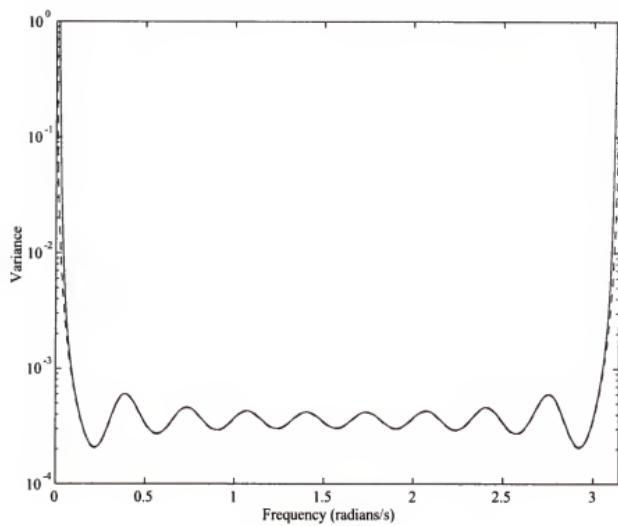


Figure 2.2: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds for a Single Sinusoid of Varying Frequency

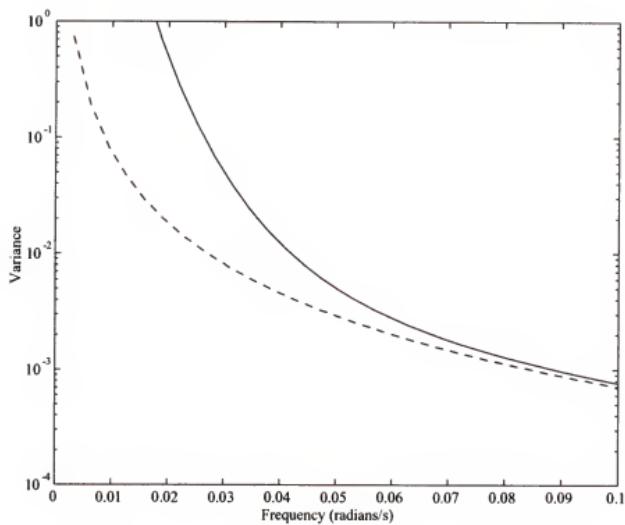


Figure 2.3: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds for a Single Sinusoid at Low Frequencies

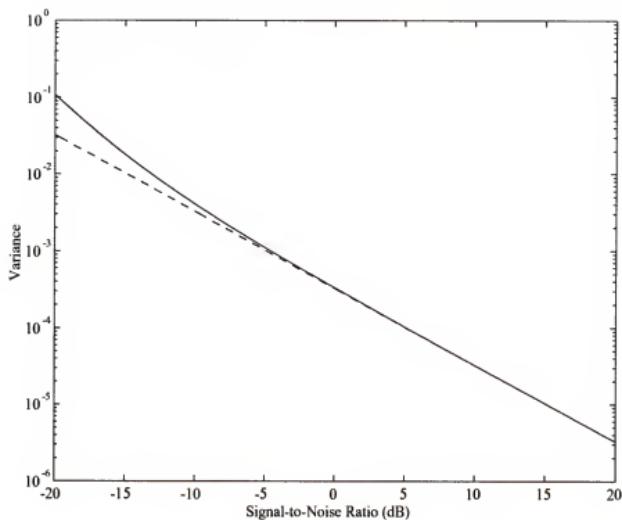


Figure 2.4: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds for a Single Sinusoid as a Function of Signal-to-Noise Ratio

the first signal to increase by a factor of more than ten; it has also caused the ratio between the two bounds to increase. Note that the separation between the sinusoids $\Delta f = 0.1/\pi = 1/31.42 \approx 1/L$, so the two signals are relatively close in frequency.

The effect of changes in the frequency separation is shown in Figure 2.6. As the frequency separation is reduced, the bounds on the variance of estimators increase dramatically. In this example, $a_1 = a_2 = 1$, $\omega_1 = 0.2$, $\phi_1 = \phi_2 = 0$, $L = 39$, $\sigma^2 = 0.1$, and ω_2 varies between 0.2 and 0.5. (The fact that the increase in the bounds occurs near the resolution limit for the classical spectral estimation techniques, $\Delta f \approx 1/L$, is coincidental; if the noise power is reduced, the increase occurs at smaller frequency separations.)

The Bhattacharyya bound derived in this chapter is a tighter bound for the variance of an unbiased estimator of the frequencies of real sinusoids than the widely known Cramér-Rao bound. The difference between the two bounds is most pronounced under three circumstances: when the number of samples is small, when the signal-to-noise ratio is low, or when signal contains multiple closely spaced real or complex sinusoids. The Bhattacharyya bound is therefore most useful in evaluating the performance of frequency estimators when these conditions apply.

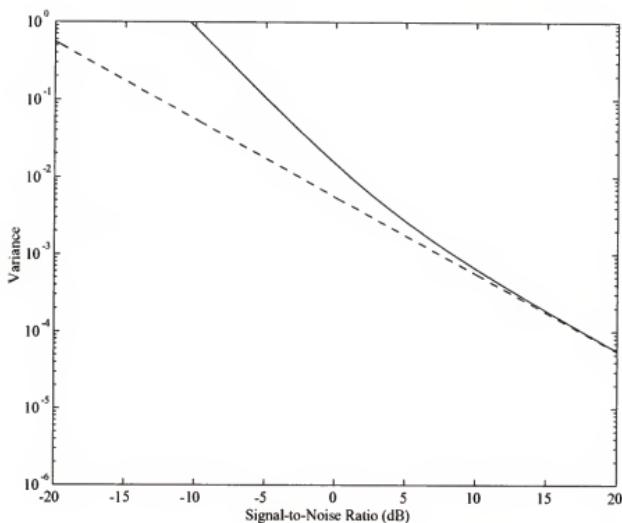


Figure 2.5: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds for One of Two Sinusoids as a Function of Signal-to-Noise Ratio

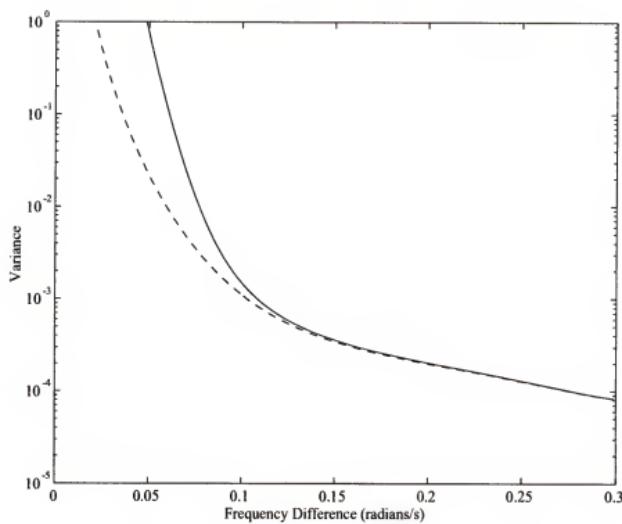


Figure 2.6: Cramér-Rao (dashed line) and Bhattacharyya (solid line) Bounds for One of Two Sinusoids as a Function of Frequency Difference

CHAPTER 3

SUBSPACE ESTIMATION TECHNIQUES

All subspace estimators are based on the properties of invariant subspaces of autocorrelation matrices. These properties give the subspace techniques their high performance, but are also responsible for their high computational cost, since finding eigenvalues or eigenvectors of a general $M \times M$ matrix requires $\mathcal{O}(M^3)$ operations. In the following section, the most widely used subspace methods for frequency estimation are described, and the computational techniques used are detailed. This will provide a framework for later discussion of fast subspace techniques.

Eigendecomposition of Exact Autocorrelation Matrices

If the autocorrelation matrix \mathbf{R} of the input signal is known exactly, the subspace techniques can determine the frequencies of sinusoids in the input signal exactly. Of course, in practice the autocorrelation matrix is not known precisely, but must be estimated from the data. Nevertheless, the principles on which the subspace techniques are based may be seen most clearly by beginning with the simplest case, where we assume exact knowledge of the autocorrelation matrix.

Although the focus of this work is on real-valued signals, in this chapter complex signals are assumed for notational convenience. For any of the expressions given here for complex signals, equivalent expressions for real signals may be found by expressing each real sinusoid as the sum of 2 complex sinusoids. The expressions for real signals that are needed in later chapters will be given as they are developed.

Noise-Free Signals

We will begin with the simplest case, where no noise is present. Consider a discrete-time signal composed of N complex sinusoids:

$$x[k] = \sum_{i=1}^N a_i \exp(j\omega_i k), \quad (3.1)$$

where $a_i \neq 0$ and $-\pi < \omega_i < \pi$ for each i , and $\omega_i \neq \omega_j$ if $i \neq j$. This is the complex form of the deterministic signal model described in Chapter 2. The elements of the autocorrelation matrix of this signal are given by

$$\begin{aligned} R_{xx}(i, j) &= \langle x[k_0 + i]^* x[k_0 + j] \rangle \\ &= x[k_0 + i]^* x[k_0 + j]. \end{aligned}$$

Because it is an autocorrelation, \mathbf{R}_{xx} is Hermitian; for real data, \mathbf{R}_{xx} is symmetric. When $x[k]$ is of the form given by (3.1), \mathbf{R}_{xx} may also be written as

$$\mathbf{R}_{xx} = \tilde{\mathbf{S}} \tilde{\mathbf{S}}^H, \quad (3.2)$$

where the columns of the $M \times N$ matrix $\tilde{\mathbf{S}}$ are defined by

$$\tilde{\mathbf{s}}_i = a_i \exp(j\omega_i k_0) \mathbf{s}_i,$$

and \mathbf{s}_i is the vector valued function (of length M)

$$\mathbf{s}_i = \mathbf{s}(\omega_i) \equiv [1, \exp(j\omega_i), \exp(j2\omega_i), \dots, \exp(j(M-1)\omega_i)]^T.$$

The autocorrelation of a noise-free signal composed of sinusoids is thus equal to the outer product of a set of N vectors of length M ; and the frequencies that define

those vectors are the same as the frequencies of the sinusoids in the signal. If the input signal is composed of N_R real sinusoids, then \mathbf{R}_{xx} can be expressed as a sum of $N = 2N_R$ complex sinusoids, so the rank of \mathbf{R}_{xx} is twice the number of real signals.

For $M > 1$ and $-\pi < \omega_1, \omega_2 < \pi$, $\mathbf{s}(\omega_1)$ and $\mathbf{s}(\omega_2)$ are linearly independent if $\omega_1 \neq \omega_2$. If $M \geq N$ and all of the ω_i are distinct, the N vectors \mathbf{s}_i will be linearly independent, and \mathbf{R}_{xx} will have rank N . An eigendecomposition of \mathbf{R}_{xx} is

$$\mathbf{R}_{xx} = \sum_{i=1}^M \lambda_i \mathbf{e}_i \mathbf{e}_i^H, \quad (3.3)$$

where we will also require that the eigenvectors are ordered so $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$. Note that since it is an autocorrelation matrix, \mathbf{R}_{xx} is positive semidefinite, so all of its eigenvalues λ_i are nonnegative. With the eigenvalues of \mathbf{R}_{xx} arranged in descending order, only the first N are nonzero.

Signals in White Noise

Now suppose that $x[k]$ is corrupted by an additive noise signal $e[k]$ that is uncorrelated with $x[k]$. The received signal is given by:

$$y[k] = \sum_{i=1}^N a_i \exp(j\omega_i k) + e[k].$$

If the noise $e[k]$ is white with variance σ_e^2 , the autocorrelation matrix \mathbf{R}_{yy} of the received signal will be

$$\mathbf{R}_{yy} = \mathbf{R}_{xx} + \sigma_e^2 \mathbf{I}. \quad (3.4)$$

Using (3.3) and the fact that the eigenvectors of \mathbf{R}_{xx} are orthogonal,

$$\mathbf{I} = \sum_{i=1}^M \mathbf{e}_i \mathbf{e}_i^H,$$

we can write (3.4) as

$$\mathbf{R}_{yy} = \sum_{i=1}^N \lambda_i \mathbf{e}_i \mathbf{e}_i^H + \sigma_e^2 \sum_{i=1}^M \mathbf{e}_i \mathbf{e}_i^H.$$

This shows that the eigenvectors of \mathbf{R}_{xx} are also eigenvectors of \mathbf{R}_{yy} , and that an eigendecomposition of \mathbf{R}_{yy} is given by:

$$\mathbf{R}_{yy} = \sum_{i=1}^N \left(\lambda_i + \sigma_e^2 \right) \mathbf{e}_i \mathbf{e}_i^H + \sum_{i=N+1}^M \sigma_e^2 \mathbf{e}_i \mathbf{e}_i^H.$$

Since the first N λ_i are all greater than zero, the N largest eigenvalues of \mathbf{R}_{yy} are greater than σ_e^2 , and the remaining $M - N$ eigenvalues are equal to σ_e^2 . The eigenvectors corresponding to the N largest eigenvalues of \mathbf{R}_{yy} are referred to as the signal eigenvectors, and the remaining eigenvectors of \mathbf{R}_{yy} are referred to as the noise eigenvectors. The invariant subspaces spanned by these two groups of eigenvectors are referred to as the signal and noise subspaces, respectively. Two important properties of this decomposition are the foundation for the subspace methods.

First, because the \mathbf{s}_i span the signal subspace and because $\mathbf{s}(\omega_i)$ and $\mathbf{s}(\omega_j)$ are linearly independent as long as $i \neq j$, the only values of $\mathbf{s}(\omega)$ that lie entirely in the signal subspace are the $\mathbf{s}(\omega_i)$. The signal and noise subspaces are orthogonal and together span the entire M dimensional space, so any vector of length M that does not lie entirely in the signal subspace has a nonzero projection on the noise subspace. This means that the only roots of $\mathbf{s}(\omega)^H \mathbf{v}$, for any vector \mathbf{v} in the noise subspace, are the original frequencies ω_i . For any other value of ω , $\mathbf{s}(\omega)$ is linearly independent of the \mathbf{s}_i , and will have a nonzero projection on the noise subspace, that is, $\mathbf{s}(\omega)^H \mathbf{v} \neq 0$ if $\omega \neq \omega_i$.

This property allows us to determine the frequencies in the input signal by finding the roots of

$$p(\omega) = \left\| \mathbf{s}(\omega)^H \mathbf{V}_N \right\|_2^2$$

for $-\pi < \omega < \pi$, where \mathbf{V}_N is any $M \times K$ matrix whose columns are in the noise subspace. Subspace estimators that rely on this property are referred to as noise subspace methods.

A second important property arises because the matrix \mathbf{R}_{yy} is Hermitian positive semidefinite. The eigenvectors of \mathbf{R}_{yy} are the same as its singular vectors, and its eigenvalues are the same as its singular values. Using the properties of the singular value decomposition, it can be shown that

$$\mathbf{R}'_{yy} = \sum_{i=1}^N (\lambda_i + \sigma_e^2) \mathbf{e}_i \mathbf{e}_i^H \quad (3.5)$$

is the best rank N approximation, in the 2-norm sense, to \mathbf{R}_{yy} [34]. \mathbf{R}'_{yy} may therefore be used as a noise-reduced approximation to \mathbf{R}_{xx} . Estimators that follow this approach are referred to as signal subspace estimators.

Signals in Colored Noise

In the discussion above, it was assumed that the noise was white. If the noise is not white, the subspace methods may still be employed, but the computation of the signal and noise subspaces is somewhat more complicated. First, it is necessary that the noise autocorrelation matrix Σ be known or estimated. Then, instead of computing invariant subspaces for the standard problem $\mathbf{R}_{yy}\mathbf{e} = \lambda\mathbf{e}$, the invariant subspaces of the generalized eigenproblem $\mathbf{R}_{yy}\mathbf{e} = \lambda\Sigma\mathbf{e}$ are computed. If Σ is nonsingular, the invariant subspaces of $\mathbf{R}_{yy}\mathbf{e} = \lambda\Sigma\mathbf{e}$ are the same as those of the modified problem $\Sigma^{-1}\mathbf{R}_{yy}\mathbf{e} = \lambda\mathbf{e}$; intuitively, we may view this as a whitening transformation, which converts the generalized eigenproblem into a standard problem.

The matrix pencil defined by \mathbf{R}_{yy} and Σ is definite, since Σ is positive definite. In addition to the simple transformation described above, several other conventional methods exist for solving the definite generalized eigenproblem. Further details on

the methods used to compute the signal and noise subspaces in the colored noise case are given later in this chapter, where the computation of the subspace methods is discussed.

Eigendecomposition of Estimated Autocorrelation Matrices

When the subspace methods are used on noisy data, the autocorrelation is not known exactly, but must be estimated from the data. The properties of the exact autocorrelation matrix described above are not true for the estimated matrix; nevertheless, useful frequency estimates may still be obtained by assuming the properties are good approximations for the estimated matrix as well. Any estimated autocorrelation matrix will have an eigendecomposition given by

$$\hat{\mathbf{R}}_{yy} = \sum_{i=1}^N \hat{\lambda}_i \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H + \sum_{i=N+1}^M \hat{\lambda}_i \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H, \quad (3.6)$$

where the estimated signal and noise subspaces are shown separately. Note that the estimated noise eigenvalues are unequal with probability one when noise is present [10]. The estimated noise subspace is different from the exact noise subspace, and will no longer be orthogonal to the signal vectors \mathbf{s}_i , so the function

$$\hat{p}(\omega) = \left\| \mathbf{s}(\omega)^H \hat{\mathbf{V}}_N \right\|_2^2 \quad (3.7)$$

where $\hat{\mathbf{V}}_N$ is any matrix with columns in the estimated noise subspace, will no longer have roots at the ω_i . (Typically, it will not have any real roots at all.) If the autocorrelation estimate is close to the actual autocorrelation, the estimated noise subspace will be a good approximation to the noise subspace of the exact autocorrelation matrix, and it will be possible to estimate the ω_i by, for example, finding the N smallest minima of (3.7) for $-\pi < \omega < \pi$. Since finding the roots of a polynomial is usually

simpler than finding minima of an arbitrary function, an alternative way of finding the minima is to locate the roots of

$$\hat{p}(z) = \left\| \mathbf{z}^H \hat{\mathbf{V}}_N \right\|_2^2, \quad (3.8)$$

where

$$\mathbf{z} \equiv [1, z, z^2, \dots, z^{M-1}]^T.$$

The N smallest minima of equation (3.7) correspond to the N roots of equation (3.8) closest to the unit circle.

If the input data are real, $\hat{\mathbf{R}}_{yy}$ is real and symmetric, and the eigenvectors of $\hat{\mathbf{R}}_{yy}$ are real. Since a real sinusoid is the sum of 2 complex sinusoids, the dimension of the signal subspace is twice the number of real signals. Because the eigenvectors are real, $\hat{p}(\omega) = \hat{p}(-\omega)$ and $\hat{p}(z) = \hat{p}(z^*)$ for any choice of $\hat{\mathbf{V}}_N$, as would be expected. This allows the search for minima to be restricted to $0 < \omega < \pi$ for real signals, and the search for roots to be restricted to the upper half-plane.

Estimating the Number of Signals

In the preceding discussion, it has been implicitly assumed that the number of signals present in the input data was known. If this is not the case, the need to estimate the number of signals complicates the use of the subspace methods. Analogous difficulties arise with other methods, and the number of signals estimation problem is the subject of ongoing research.

In some circumstances, determination of the number of signals is simple: if the signal-to-noise ratio is high, then there will be a gap between the magnitudes of the signal and noise eigenvectors in (3.6), that will indicate the number of signals.

At low signal-to-noise ratios, however, it becomes difficult to determine the number of signals in this manner.

One approach is to perform likelihood ratio tests on the eigenvalues of $\hat{\mathbf{R}}_{yy}$. Although this is a straightforward approach, it is complicated by the need to set a threshold for accepting the hypothesis that a certain number of signals are present; this introduces a subjective element into the process, since the choice of threshold is arbitrary. Drawing on information theoretic arguments, several authors [11–14] have proposed alternatives that do not require selection of a threshold. These approaches use an estimator of the form

$$f(\max_{\Theta} P_n(\Theta)) - p(n),$$

where $f(\max_{\Theta} P_n(\Theta))$ is some function of the maximum value of the likelihood function for a model with n signals $P_n(\Theta)$, and $p(n)$ is a penalty function, which balances the improvement in likelihood with increasing number of signals against a cost attributed to increasing model complexity. The two most widely used variants of this approach are those due to Akaike [11], (the Akaike information criterion, or AIC), and to Schwartz and Rissanen [12, 13] (the minimum description length criterion, or MDL). In these two cases, the estimated number of signals is given by the value of n that minimizes the functions

$$\begin{aligned} AIC(n) &= -2 \log(\max_{\Theta} P_n(\Theta)) + 2N_P, \\ MDL(n) &= -\log(\max_{\Theta} P_n(\Theta)) + \frac{1}{2}N_P \log L, \end{aligned}$$

where N_P is the number of parameters in the model used to calculate the likelihood, and L is the number of samples used to estimate the model parameters.

The major difficulty in applying these techniques to the frequency estimation problem is the difficulty of calculating the maximum likelihood for the deterministic signal model, as discussed in Chapter 2. The calculation of the maximum likelihood for a single value of n requires the use of numerical maximization procedures to search for a maximum of the likelihood function, and the fact that the likelihood function is multimodal requires the use of a high precision frequency estimation technique to generate a starting value for the maximization. Often, subspace estimators are used for this purpose. It is clearly impractical to attempt to use the exact maximum likelihood to estimate the number of signals with the AIC or MDL, in the context of a subspace estimation procedure, since calculating the maximum for just one value of n requires more computation than the entire subspace frequency estimation.

An alternative strategy is to employ a different signal model for which the likelihood is more easily calculated. The stochastic signal model, which assumes the signal is stationary and Gaussian, has been widely used for this purpose. Using this model, Wax and Kailath [15, 16] developed forms of the AIC and MDL estimators suitable for use in subspace estimation. These estimators are both based on a measure α of the inequality of the noise subspace eigenvalues:

$$\alpha = \frac{\left(\prod_{i=n+1}^M \tilde{\lambda}_i \right)^{1/(M-n)}}{\frac{1}{M-n} \sum_{i=n+1}^M \tilde{\lambda}_i}, \quad (3.9)$$

where the $\tilde{\lambda}_i$ are the maximum likelihood estimates of the eigenvalues of \mathbf{R}_{yy} . Wax [16] shows that, under the stochastic signal model, these are equal to the eigenvalues $\hat{\lambda}_i$ of the covariance estimate of the autocorrelation matrix.

If the input signals are complex, the estimators are:

$$AIC_C(n) = -2L(M-n) \log \alpha + 2n(2M-n+1), \quad (3.10)$$

$$MDL_C(n) = -L(M-n) \log \alpha + \frac{1}{2}n(2M-n+1) \log L. \quad (3.11)$$

where M is the dimension of the autocorrelation matrix, and L is the number of points in the time series from which the matrix was derived. (Note that the versions of equations (3.10) and (3.11) given in [15, 16] contain a typographical error in the last term, which has been corrected here; the corrected expressions are also given in [17].) For real signals, the number of model parameters should decrease, because the eigenvectors are real; however, the resulting mismatch between the likelihood term and the penalty function sometimes causes poor performance in practice, so the complex signal form is generally preferred.

Since each real signal is associated with two eigenvalues of the autocorrelation matrix, it is reasonable to restrict the values of n in the real signals case to multiples of 2. If the number of real signals is n_R , the estimators are

$$AIC_R(n_R) = -2L(M-2n_R) \log \alpha + 4n_R(2M-2n_R+1), \quad (3.12)$$

$$MDL_R(n_R) = -L(M-2n_R) \log \alpha + n_R(2M-2n_R+1) \log L. \quad (3.13)$$

The estimated number of signals \hat{N} for either method is the value of n that minimizes $AIC(n)$ or $MDL(n)$. It has been shown [15] that the AIC is not a consistent estimator of the actual number of signals, while the MDL is; this is in agreement with empirical findings that the AIC tends to overestimate the number of sinusoidal signals in a data record.

Although subsequent work [18, 19] has developed versions of the MDL that are applicable for a wider range of input signals, there is no entirely satisfactory method for estimating the number of signals. Since the primary focus of this work is on the computational efficiency of subspace methods, rather than the number of signals estimation problem, we will restrict our attention to developing techniques

for computing the AIC and MDL with the fast subspace algorithms developed later. These techniques are described in Chapter 8.

Subspace Methods

Although many different subspace methods have been proposed, and each has its own advantages and disadvantages, only a few techniques have achieved widespread application. To illustrate the similarities between the subspace techniques, and to provide a framework for later discussion of fast subspace methods, we will briefly describe the first subspace technique, the Pisarenko harmonic decomposition, and then discuss the three most widely used techniques: the principal components method, MUSIC, and ESPRIT.

After the discussion of each technique, a brief listing of the computational steps for calculating frequency estimates is given. The algorithms employ a mixture of conventional mathematical notation and MATLAB-style notation. In MATLAB notation, $i:j$ is used to refer to a range of array indices, so that $a(3:6)$ is a vector composed of elements 3 through 6 of a , and $B(1:2, 1:3)$ is the leading 2×3 submatrix of \mathbf{B} .

The Pisarenko Harmonic Decomposition

The Pisarenko harmonic decomposition [20] was the first of the subspace techniques to be proposed. It assumes that the number of signals is known, and that the dimension M of the autocorrelation matrix is chosen to be one more than the number of signals (or $M = 2N_R + 1$ for real signals). The estimated noise subspace is therefore spanned by a single eigenvector $\hat{\mathbf{e}}_M$. Then, using (3.7) or (3.8), with $\hat{\mathbf{V}} = \hat{\mathbf{e}}_M$, the estimated frequencies $\hat{\omega}_i$ are found. Note that there is no ambiguity in the selection of minima or roots, since the polynomial has at most N minima on the unit circle or roots in the complex plane, the same as the number of signals.

The statistical properties of the Pisarenko harmonic decomposition have been examined by Stoica and Nehorai [21]. The technique is asymptotically unbiased, but statistically inefficient, that is, the variances of the frequency estimates do not approach the Cramér-Rao bounds as the number of samples increases. In practice, the performance of the Pisarenko method is inferior to other subspace techniques; its principal advantage is its simplicity.

ALGORITHM 3.1: PISARENKO HARMONIC DECOMPOSITION

```

Determine the number of signals  $N$ 
Estimate the  $N + 1 \times N + 1$  signal autocorrelation matrix  $\hat{\mathbf{R}}$ 
if noise is colored
    Estimate the  $N + 1 \times N + 1$  noise autocorrelation matrix  $\hat{\Sigma}$ 
else
     $\hat{\Sigma} = \mathbf{I}$ 
end
Compute the eigenvector  $\hat{\mathbf{e}}_{N+1}$  associated with the
smallest eigenvalue of  $\hat{\mathbf{R}}\hat{\mathbf{e}} = \lambda\hat{\Sigma}\hat{\mathbf{e}}$ 
 $\hat{p}(z) = [1, z^{-1}, z^{-2}, \dots, z^{-N}]\hat{\mathbf{e}}_{N+1}$ 
Find the  $N$  roots  $z_1 \dots z_N$  of  $\hat{p}(z)$ 
for  $i = 1 : N$ 
     $\hat{\omega}_i = \text{angle}(z_i)$ 
end

```

The Principal Components Method

The principal components method [22–24] of Kumaresan and Tufts is a signal subspace method that uses a reduced rank approximation to $\hat{\mathbf{R}}_{yy}$ to solve the linear prediction equations. From equation (3.5), the rank N matrix that is closest in the 2-norm to $\hat{\mathbf{R}}_{yy}$ is

$$\hat{\mathbf{R}}_{PC} = \sum_{i=1}^N \hat{\lambda}_i \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H. \quad (3.14)$$

Note here that the signal subspace should be chosen so that $\hat{\mathbf{R}}_{PC}$ is always positive definite, since negative eigenvalues can only arise due to the error in $\hat{\mathbf{R}}_{yy}$, and should

only be associated with the noise subspace. The reduced-rank matrix $\hat{\mathbf{R}}_{PC}$ is used to solve the Yule-Walker equation

$$\hat{\mathbf{R}}_{PC}\mathbf{a} = -\mathbf{r} \quad (3.15)$$

Since the solution to (3.15) is not unique, the solution is chosen that minimizes the length of \mathbf{a} . This may be efficiently determined by using the properties of the singular value decomposition and the fact that the SVD of $\hat{\mathbf{R}}_{PC}$ is given by (3.14), since $\hat{\mathbf{R}}_{PC}$ is positive semidefinite:

$$\mathbf{a} = -\hat{\mathbf{R}}_{PC}^+\mathbf{r} = -\left(\sum_{i=1}^N \frac{1}{\lambda_i} \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H\right) \mathbf{r} \quad (3.16)$$

(where $\hat{\mathbf{R}}_{PC}^+$ is the pseudoinverse of $\hat{\mathbf{R}}_{PC}$). The N roots of

$$\left[1, z^{-1}, z^{-2}, \dots, z^{-M}\right] \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} \quad (3.17)$$

closest to the unit circle are used to find the principal components estimates of the frequencies ω_i . It has been shown [23, 24] that the use of a rank reduced matrix $\hat{\mathbf{R}}_{PC}$ with $N < M$ causes the polynomial (3.17) to have N roots very close to the unit circle, with the remaining roots clustered well inside the unit circle, simplifying identification of the desired roots.

In comparison with MUSIC and ESPRIT, less analytical work has been performed on the statistical properties of the principal components method. Simulation results [22] indicate that the technique has a higher threshold signal-to-noise ratio than MUSIC, but otherwise similar performance, and an analytical comparison between the two techniques [25] yields the same conclusion.

ALGORITHM 3.2: PRINCIPAL COMPONENTS METHOD

```

Estimate the  $M \times M$  signal autocorrelation matrix  $\hat{\mathbf{R}}$ 
if noise is colored
  Estimate the  $M \times M$  noise autocorrelation matrix  $\hat{\Sigma}$ 
else
   $\hat{\Sigma} = \mathbf{I}$ 
end
if number of signals  $N$  is unknown
  Compute the eigenvalues of  $\hat{\mathbf{R}}\hat{\mathbf{e}} = \hat{\lambda}\hat{\Sigma}\hat{\mathbf{e}}$ 
  Estimate the number of complex signals  $N$ , using the AIC or MDL
end
Compute the  $N$  largest eigenvalues and the associated eigenvectors
 $\mathbf{a} = -\sum_{i=1}^N \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^T \mathbf{r} / \hat{\lambda}_i$ 
 $\hat{p}(z) = 1 + \sum_{i=1}^M z^{-i} a(i)$ 
Find the  $N$  roots  $z_1 \dots z_N$  of  $\hat{p}(z)$  closest to the unit circle
for  $i = 1 : N$ 
   $\hat{\omega}_i = \text{angle}(z_i)$ 
end

```

The MUSIC Algorithm

MUSIC (an acronym for MULTiple SIgnal Classification) was developed by Schmidt [10, 26] in the late 1970's. For uniformly sampled time series, MUSIC is a straightforward generalization of the Pisarenko harmonic decomposition. Rather than a single vector, it employs a matrix composed of several vectors from the noise subspace,

$$\hat{\mathbf{E}}_N = [\hat{\mathbf{e}}_{N+1}, \dots, \hat{\mathbf{e}}_M]$$

and uses (3.7) or (3.8) to find the MUSIC estimates of the ω_i . (The root-finding variant (3.8) is referred to as root-MUSIC, and was first suggested by Barabell [27].)

Since there are generally many more eigenvectors in the noise subspace than in the signal subspace, it is important to note that use of the identity

$$\sum_{i=1}^N \hat{\mathbf{e}}_i^H \hat{\mathbf{e}}_i + \sum_{i=N+1}^M \hat{\mathbf{e}}_i^H \hat{\mathbf{e}}_i = \mathbf{I} \quad (3.18)$$

allows computation of whatever set of eigenvectors is smaller, and may greatly reduce the amount of computation required to compute noise subspace estimators:

$$\hat{p}(z) = \left\| \mathbf{z}^H \hat{\mathbf{E}}_N \right\|_2^2 = \mathbf{z}^H \left(\sum_{i=N+1}^M \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H \right) \mathbf{z} = \mathbf{z}^H \left(\mathbf{I} - \sum_{i=1}^N \hat{\mathbf{e}}_i \hat{\mathbf{e}}_i^H \right) \mathbf{z}.$$

The statistical properties of MUSIC have been widely studied [2, 8, 28–31]. It has been shown [2] that MUSIC is asymptotically efficient, and that it is an approximate maximum likelihood estimator for large data records. Surprisingly, it has also been shown that MUSIC can in some circumstances have a lower variance than the maximum likelihood estimator [8], which is possible since the MLE is not efficient for a finite number of samples. Stoica has analyzed the performance of MUSIC and ESPRIT for the estimation of sinusoidal parameters [29], with ESPRIT found to be slightly superior in terms of minimizing the mean squared error of the frequency estimates. A companion paper [32], which analyzed the two techniques for array bearing estimation, found MUSIC to be superior for that application.

ALGORITHM 3.3: ROOT-MUSIC

```

Estimate the  $M \times M$  signal autocorrelation matrix  $\hat{\mathbf{R}}$ 
if noise is colored
    Estimate the  $M \times M$  noise autocorrelation matrix  $\hat{\Sigma}$ 
else
     $\hat{\Sigma} = \mathbf{I}$ 
end
if number of signals  $N$  is unknown
    Compute the eigenvalues of  $\hat{\mathbf{R}}\hat{\mathbf{e}} = \hat{\lambda}\hat{\Sigma}\hat{\mathbf{e}}$ 
    Estimate the number of complex signals  $N$ , using the AIC or MDL
end
if  $N < M/2$ 
    Compute the eigenvectors associated with the  $N$  largest eigenvalues
     $\hat{\mathbf{E}}_S = [\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_N]$ 
     $\mathbf{U} = \mathbf{I} - \hat{\mathbf{E}}_S \hat{\mathbf{E}}_S^H$ 
else
    Compute the eigenvectors associated with the  $M - N$  smallest eigenvalues
     $\hat{\mathbf{E}}_N = [\hat{\mathbf{e}}_{N+1}, \dots, \hat{\mathbf{e}}_M]$ 
     $\mathbf{U} = \hat{\mathbf{E}}_N \hat{\mathbf{E}}_N^H$ 
end
 $\hat{p}(z) = \mathbf{z}^H \mathbf{U} \mathbf{z}$ , where  $\mathbf{z} = [1, z, z^2, \dots, z^{M-1}]^T$ 
Find the  $N$  roots  $z_1 \dots z_N$  of  $\hat{p}(z)$  closest to the unit circle
for  $i = 1 : N$ 
     $\hat{\omega}_i = \text{angle}(z_i)$ 
end

```

The ESPRIT Algorithm

ESPRIT (Estimation of Signal Parameters via Rotational Invariance Techniques) was developed in the mid-1980's by Roy and Kailath [33–35]. As its name implies, it is based on a different principle than other subspace methods, namely, the invariance of the signal subspace to time shifts. We have shown above that the signal

subspace is spanned by the N vectors \mathbf{s}_i ; that is, the columns of the matrix

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N] = \begin{bmatrix} 1 & 1 & \dots & 1 \\ e^{j\omega_1} & e^{j\omega_2} & \dots & e^{j\omega_N} \\ e^{j2\omega_1} & e^{j2\omega_2} & \dots & e^{j2\omega_N} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j(M-1)\omega_1} & e^{j(M-1)\omega_2} & \dots & e^{j(M-1)\omega_N} \end{bmatrix}$$

form a basis for the signal subspace. Consider the two submatrices of \mathbf{S} composed of its first $M - 1$ rows, and its last $M - 1$ rows,

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 \\ \times \times \times \end{bmatrix} = \begin{bmatrix} \times \times \times \\ \mathbf{S}_2 \end{bmatrix}. \quad (3.19)$$

These matrices are related by $\mathbf{S}_2 = \mathbf{S}_1 \mathbf{Q}$, where

$$\mathbf{Q} = \begin{bmatrix} e^{j\omega_1} & 0 & \dots & 0 \\ 0 & e^{j\omega_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{j\omega_N} \end{bmatrix}.$$

This relationship between the two submatrices is a consequence of the fact that each of the signals is invariant under a time shift, except for multiplication by a constant. The choice of \mathbf{S}_1 and \mathbf{S}_2 given above selects the two submatrices that have maximum overlap; although this is the most common choice, and appears to give the best performance for frequency estimation, many other choices are possible. A generalized variant of ESPRIT that attempts to exploit all possible invariances in the context of array processing is described in [36].

The set of signal eigenvectors that are the columns of \mathbf{E} spans the same space as the columns of \mathbf{S} , so the two matrices must be related by a nonsingular transformation $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N] = \mathbf{S}\mathbf{W}$. Partitioning \mathbf{E} in the same manner as \mathbf{S} and using (3.19),

$$\mathbf{E} = \begin{bmatrix} \mathbf{E}_1 \\ \times \times \times \\ \mathbf{E}_2 \end{bmatrix} = \begin{bmatrix} \times \times \times \\ \mathbf{S}_1 \mathbf{W} \\ \times \times \times \\ \mathbf{S}_2 \mathbf{W} \end{bmatrix} = \begin{bmatrix} \times \times \times \\ \mathbf{S}_2 \mathbf{W} \end{bmatrix} = \mathbf{S}\mathbf{W}.$$

This means that the submatrices of \mathbf{E} are also related by a nonsingular transformation $\mathbf{E}_2 = \mathbf{S}_2 \mathbf{W} = \mathbf{S}_1 \mathbf{Q} \mathbf{W} = \mathbf{E}_1 \mathbf{W}^{-1} \mathbf{Q} \mathbf{W}$, or

$$\mathbf{E}_1 \mathbf{W}^{-1} \mathbf{Q} \mathbf{W} = \mathbf{E}_1 \mathbf{X} = \mathbf{E}_2. \quad (3.20)$$

Since \mathbf{W} must be nonsingular, \mathbf{X} and \mathbf{Q} are similar, and therefore the eigenvalues of \mathbf{X} are the same as the eigenvalues of \mathbf{Q} . The arguments of the eigenvalues of \mathbf{Q} are the original frequencies, so if \mathbf{X} is found, the frequencies can be easily determined.

When an estimate of the autocorrelation matrix is used, the partitions of the matrix of signal subspace eigenvectors will not exactly satisfy (3.20). An approximate solution must be sought, and the most common approach is to solve (3.20) in the total least squares sense, that is, to find an \mathbf{X}_{TLS} that solves

$$(\hat{\mathbf{E}}_1 + \mathbf{F}_1) \mathbf{X}_{TLS} = (\hat{\mathbf{E}}_2 + \mathbf{F}_2)$$

so that the Frobenius norm of the error matrices

$$\|[\mathbf{F}_1 \ \mathbf{F}_2]\|_F$$

is minimized. This approach is referred to as TLS-ESPRIT. If (3.20) is solved in the least squares sense, the method is sometimes called LS-ESPRIT; the TLS approach yields superior estimates. The total least squares problem may be solved using the

singular value decomposition, as described in [37]. When \mathbf{X}_{TLS} has been found, the ESPRIT frequency estimates are taken to be the arguments (angles) of the eigenvalues of \mathbf{X}_{TLS} .

The statistical properties of ESPRIT have been heavily analyzed [29, 32–35, 38]. It has been shown [34, 35] that ESPRIT is an approximate maximum likelihood estimator, and that it is consistent and asymptotically efficient. For estimation of sinusoidal frequencies, an analysis by Stoica [29] shown that ESPRIT (referred to in that work as SURE) is slightly more accurate than MUSIC in most cases.

ALGORITHM 3.4: TLS-ESPRIT

```

Estimate the  $M \times M$  signal autocorrelation matrix  $\hat{\mathbf{R}}$ 
if noise is colored
    Estimate the  $M \times M$  noise autocorrelation matrix  $\hat{\Sigma}$ 
else
     $\hat{\Sigma} = \mathbf{I}$ 
end
if number of signals  $N$  is unknown
    Compute the eigenvalues of  $\hat{\mathbf{R}}\hat{\mathbf{e}} = \hat{\lambda}\hat{\Sigma}\hat{\mathbf{e}}$ 
    Estimate the number of complex signals  $N$ , using the AIC or MDL
end
Compute the eigenvectors associated with the  $N$  largest eigenvalues
 $\hat{\mathbf{E}} = [\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_N]$ 
 $\hat{\mathbf{E}}_1 = \hat{\mathbf{E}}(1 : M - 1, 1 : N)$ 
 $\hat{\mathbf{E}}_2 = \hat{\mathbf{E}}(2 : M, 1 : N)$ 
Solve  $\hat{\mathbf{E}}_1\mathbf{X} = \hat{\mathbf{E}}_2$  in the total least squares sense
Find the  $N$  eigenvalues  $\sigma_1 \dots \sigma_N$  of  $\mathbf{X}$ 
for  $i = 1 : N$ 
     $\hat{\omega}_i = \text{angle}(\sigma_i)$ 
end

```

Computational Considerations

In subsequent chapters, new algorithms for Toeplitz eigendecomposition will be developed and employed to reduce the computation required for frequency estimation

using subspace techniques. To provide a context for evaluating these improvements, we will briefly examine the computational cost, in terms of operation counts and memory requirements, of the subspace techniques when they are implemented using conventional methods. All of the computational costs given will be for the real signals case, since real signals are assumed in the remainder of this work.

We can see immediately that all of the conventional implementations require $\mathcal{O}(M^2)$ memory locations; the amount of computation required depends on the particular algorithm used, and on whether the noise is white or colored. The cost of each of the three most widely used algorithms is discussed in more detail below.

Estimating frequencies with a subspace algorithm requires three basic computational steps: estimate the signal and noise correlation matrices, compute some of the eigenvectors (an invariant subspace) of the matrix pencil $(\hat{\mathbf{R}}, \hat{\Sigma})$, and process this invariant subspace in some fashion to determine the frequencies. If there are L points in the input data record, N sinusoidal signals, and an $M \times M$ autocorrelation matrix is used, then estimation of the correlation matrices is typically $\mathcal{O}(LM)$, computation of the invariant subspace is $\mathcal{O}(M^3)$, and estimation of the frequencies from the invariant subspace is approximately $\mathcal{O}(M^2N)$ for the principal components method and MUSIC, and $\mathcal{O}(MN^2)$ for ESPRIT. Since M is larger than N , and is typically one-tenth to one-half the size of L , the computational cost of conventional methods is dominated by the computation of the invariant subspace.

The cost of invariant subspace computation varies depending on whether the noise is white or colored. In mathematical terms, the white noise problem requires computation of an invariant subspace of a symmetric matrix, and the colored noise problem requires computing an invariant subspace of a definite matrix pencil.

White Noise

When conventional techniques for symmetric eigenproblem (Householder reduction to tridiagonal form, followed by QR or QL iteration) are used, approximately $4M^3/3$ floating-point operations are required to compute the eigenvalues for the white noise case. Computing both the eigenvalues and the eigenvectors by the same method requires approximately $9M^3$ floating-point operations, because each of the orthogonal transformations used to reduce $\hat{\mathbf{R}}$ to tridiagonal form must be accumulated for use in computing the eigenvectors. If fewer than 23 eigenvectors are required, it is more economical to compute only the eigenvalues with the QL algorithm, since the eigenvectors may then be computed by inverse iteration at a cost of $M^3/3$ operations each.

Inverse “iteration” with accurate eigenvalues typically requires only one solution of the equation $(\hat{\mathbf{R}} - \lambda_i \mathbf{I})\mathbf{x}_i = \mathbf{z}$, with \mathbf{z} initialized to a random vector, to produce an accurate eigenvector \mathbf{x}_i . Inverse iteration will fail if \mathbf{z} is orthogonal to \mathbf{x}_i , but this is extremely unlikely for random \mathbf{z} ; difficulties may also arise when $\hat{\mathbf{R}}$ has repeated eigenvalues, because inverse iteration will find only one of the eigenvectors associated with the repeated values. When $\hat{\mathbf{R}}$ is a matrix calculated from noisy data, however, repeated eigenvalues are also extremely unlikely. In most cases, then, inverse iteration is a viable alternative to accumulating the transformations in the QR or QL algorithms, as long as the dimension of the required subspace is small.

Colored Noise

In the colored noise case, several computational approaches are possible. We have already described one technique, which computes the eigenvalues of $\hat{\Sigma}^{-1}\hat{\mathbf{R}}$; it is rarely used because other methods have better numerical properties and equal computational cost. One common approach is to compute the Cholesky factorization $\hat{\Sigma} = \mathbf{G}^T \mathbf{G}$, where \mathbf{G} is upper triangular, and then compute the eigenvectors and

eigenvalues of $\mathbf{G}^{-T}\hat{\mathbf{R}}\mathbf{G}^{-1}$. This transforms the generalized eigenvalue problem into a standard eigenvalue problem, where the multiplication by \mathbf{G}^{-T} and \mathbf{G}^{-1} may also be viewed as a whitening transformation:

$$\begin{aligned}\mathbf{G}^{-T}\hat{\mathbf{R}}\mathbf{G}^{-1}\mathbf{y} &= \lambda\mathbf{G}^{-T}\hat{\Sigma}\mathbf{G}^{-1}\mathbf{y} \\ &= \lambda\mathbf{G}^{-T}\mathbf{G}^T\mathbf{G}\mathbf{G}^{-1}\mathbf{y} \\ &= \lambda\mathbf{y}\end{aligned}$$

The eigenvalues of $\mathbf{G}^{-T}\hat{\mathbf{R}}\mathbf{G}^{-1}\mathbf{y} = \lambda\mathbf{y}$ are the same as those of the original problem $\hat{\mathbf{R}}\mathbf{x} = \lambda\hat{\Sigma}\mathbf{x}$, and the eigenvectors of the original problem are given by $\mathbf{x} = \mathbf{G}^{-1}\mathbf{y}$. The computational cost is therefore the cost of solving the standard eigenproblem, plus an additional $M^3/3$ operations for the Cholesky decomposition, $M^3/3$ for inverting \mathbf{G} , $2M^3$ for computing $\mathbf{G}^{-T}\hat{\mathbf{R}}\mathbf{G}^{-1}$, and $M^3/3$ per eigenvector for multiplying by \mathbf{G}^{-1} , for a total of $(N + 5)M^3/3$ additional operations.

The common denominator of all the subspace estimators described above is the requirement to compute the eigendecomposition of the estimated autocorrelation matrix. When conventional techniques are used, the computational cost of this computation is proportional to M^3 for a $M \times M$ matrix; this is true even when, as is often the case with the subspace methods, only a small set of eigenvectors must be computed. In Chapter 6, we will examine methods that compute any desired set of N eigenvalues of a Toeplitz matrix and their associated eigenvectors, at a cost which is proportional to NM^2 or $NM \log^2 M$, depending on the choice of technique.

The price required for this dramatic reduction in computation is a restriction on the form of the autocorrelation matrix estimators which may be used. To determine the effect of this restriction on the performance of the subspace techniques, a detailed analysis of the accuracy of several widely used estimators, and the effect of errors

in estimating the autocorrelation on the invariant subspaces of the autocorrelation matrix, is presented in the next chapter.

CHAPTER 4

AUTOCORRELATION ESTIMATES FOR SUBSPACE METHODS

In the preceding chapter, existence of an estimate of the autocorrelation matrix of the input signal was assumed; in this chapter, we will examine some common techniques for computing this estimate. Because the deterministic signal model is nonstationary and nonergodic, the issue of autocorrelation matrix estimation is more complex than it is for the usual case of stationary random processes. Since the fast eigendecomposition techniques that are developed in later chapters require a Toeplitz matrix, we will be particularly interested in the effects of using Toeplitz estimates of the autocorrelation matrix.

The first important issue in estimating the autocorrelation of deterministic sinusoids is a consequence of the nonstationarity of the signals. From this point onward, we will return to the use of real, rather than complex, signals. Under the deterministic signal model discussed in Chapter 2, the data $y[k]$ are assumed to be of the form

$$y[k] = x[k] + n[k],$$

where

$$x[k] = \sum_{i=1}^N a_i \cos(\omega_i k + \phi_i),$$

and the noise $n[k]$ is Gaussian, stationary and uncorrelated with $x[k]$, with zero mean and covariance matrix Σ . The output of this model is not wide-sense stationary, and therefore it is also not ergodic in the wide sense, that is, the statistical expected value

of $y[k] y[k + n]$ is not equal to its time average over an infinite interval. Since the process is nonstationary, its statistical autocorrelation matrix is not Toeplitz.

The derivation of the subspace methods given in Chapter 3 reflects this fact; the subspace methods were derived using the autocorrelation of a deterministic signal. To use a Toeplitz matrix in subspace estimation techniques, it is first necessary to show that a Toeplitz matrix exists that yields exact frequency estimates; if such a matrix does not exist, then fast subspace methods based on Toeplitz estimates can never equal the performance of the original (covariance-based) subspace methods.

One common approach is to simply redefine the signal model so that the signal is stationary. If ϕ is taken to be a random variable uniformly distributed between 0 and 2π , the resulting signal is stationary (but not ergodic or Gaussian). Under this signal model, which we will refer to as the semi-deterministic model, the autocorrelation matrix of a signal composed of N sinusoidal signals in additive noise has elements given by

$$R_{SD}(i, j) = \frac{1}{2} \sum_{i=1}^N a_i^2 \cos(\omega_i |i - j|) + \Sigma(i, j).$$

The use of a model with random phase implies that the frequency estimation is performed without taking advantage of any phase information that may become available. Since the Cramér-Rao bounds for frequency estimation are asymptotically decoupled from the phase, as shown in Chapter 2, it is possible for a Toeplitz estimator based on this model to have nearly equal performance to the covariance estimator for long data records.

By examining the behavior of the conventional methods as the number of samples becomes large, we can see how this may be achieved. In the following sections, we will show that when the input consists of deterministic sinusoids in stationary, additive Gaussian noise, all the estimation methods considered here, including the

covariance method on which the conventional subspace algorithms are based, converge to the same Toeplitz matrix as the number of data points L goes to infinity. To distinguish this time-averaged “autocorrelation matrix,” which is derived from a single time series, from the (non-Toeplitz) statistical autocorrelation matrix obtained from the entire ensemble of possible time series, we will refer to the autocorrelation obtained by time averaging as the temporal autocorrelation, and denote it by $\tilde{\mathbf{R}}$. It will also be shown in the following section that the temporal autocorrelation is equal to the statistical autocorrelation matrix \mathbf{R}_{SD} of the semi-deterministic signal model with the same amplitude and frequency parameters as the deterministic model.

Both MUSIC and ESPRIT have been shown to be asymptotically efficient [2,38], that is, as the number of samples becomes large, the estimation error of the techniques approaches the Cramér-Rao bound. As the number of samples becomes large, the covariance estimate of the autocorrelation approaches a Toeplitz matrix, and the error in the frequency estimate approaches the minimum possible error. If it is possible to find a Toeplitz autocorrelation estimate that is a good approximation to the temporal autocorrelation, it is then possible to use this Toeplitz matrix in the subspace methods without a significant loss of accuracy.

The Temporal Autocorrelation Matrix

Before discussing the various methods of estimating the correlation matrix, it will be useful to derive the exact temporal autocorrelation matrix for sinusoidal signals in noise. If the exact temporal autocorrelation $\tilde{\mathbf{R}}$ is used in an asymptotically efficient subspace estimator, an exact frequency estimate results. Because of this, $\tilde{\mathbf{R}}$ can serve as a standard for evaluating the accuracy of the various estimators, and allow us to bound the error in the subspace estimates derived from an estimated autocorrelation matrix.

Using the deterministic signal model, the elements of the exact temporal autocorrelation matrix are given by

$$\tilde{R}(i, j) = \lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \sum_{k=0}^{L-1} y[i+k]y[j+k] \right\}. \quad (4.1)$$

By expanding $y[k]$ as the sum of the signal and the noise, we can write this as

$$\begin{aligned} \tilde{R}(i, j) &= \lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \left(\sum_{k=0}^{L-1} x[i+k]x[j+k] \right. \right. \\ &\quad + \sum_{k=0}^{L-1} n[i+k]x[j+k] \\ &\quad + \sum_{k=0}^{L-1} x[i+k]n[j+k] \\ &\quad \left. \left. + \sum_{k=0}^{L-1} n[i+k]n[j+k] \right) \right\}. \end{aligned} \quad (4.2)$$

Because $x[k]$ and $n[k]$ are uncorrelated and zero-mean, the limits of the two cross terms involving $x[k]$ and $n[k]$ are zero, and the elements of the exact autocorrelation matrix are

$$\begin{aligned} \tilde{R}(i, j) &= \lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \sum_{k=0}^{L-1} x[i+k]x[j+k] + n[i+k]n[j+k] \right\} \\ &= \lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \sum_{k=0}^{L-1} x[i+k]x[j+k] \right\} + \Sigma(i, j) \end{aligned}$$

(because the noise Σ is stationary). Since $\Sigma(i, j)$ is assumed to be known, $\tilde{\mathbf{R}}$ may be determined by finding the limit of the sum

$$\begin{aligned} &\frac{1}{L} \sum_{k=0}^{L-1} x[i+k]x[j+k] \\ &= \frac{1}{L} \sum_{k=0}^{L-1} \sum_{m=1}^N \sum_{n=1}^N a_m a_n \cos(\omega_m(i+k) + \phi_m) \cos(\omega_n(j+k) + \phi_n) \end{aligned}$$

$$= \frac{1}{L} \sum_{m=1}^N \sum_{n=1}^N a_m a_n \sum_{k=0}^{L-1} \cos(\omega_m(i+k) + \phi_m) \cos(\omega_n(j+k) + \phi_n)$$

as L goes to infinity. The critical element here is the inner sum over k . To simplify this sum, we will use the fact that, for $0 \leq \alpha < 2\pi$,

$$\lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \sum_{k=0}^{L-1} \cos(\alpha k + \beta) \right\} = \begin{cases} \frac{1}{2} \cos(\beta), & \alpha = 0 \\ 0, & \alpha \neq 0 \end{cases}$$

By bringing the limit and the coefficient $1/L$ inside the first two summations, and expanding the product of cosines as a sum of cosines, we find that the limit of the inner summation is given by

$$\begin{aligned} & \lim_{L \rightarrow \infty} \left\{ \frac{1}{L} \sum_{k=0}^{L-1} \cos(\omega_m(i+k) + \phi_m) \cos(\omega_n(j+k) + \phi_n) \right\} \\ &= \lim_{L \rightarrow \infty} \left\{ \frac{1}{2L} \sum_{k=0}^{L-1} \cos(\omega_m(i+k) - \omega_n(j+k) + (\phi_m - \phi_n)) \right. \\ & \quad \left. + \frac{1}{2L} \sum_{k=0}^{L-1} \cos(\omega_m(i+k) + \omega_n(j+k) + (\phi_m + \phi_n)) \right\} \\ &= \lim_{L \rightarrow \infty} \left\{ \frac{1}{2L} \sum_{k=0}^{L-1} \cos(\omega_m i - \omega_n j + (\omega_m - \omega_n)k + (\phi_m - \phi_n)) \right\} \\ &= \begin{cases} \frac{1}{2} \cos(\omega_m(i-j) + (\phi_m - \phi_n)), & \omega_m = \omega_n \\ 0, & \omega_m \neq \omega_n \end{cases} \end{aligned}$$

Since we have assumed that the sinusoids in the signal are distinct, if $\omega_m = \omega_n$, then m must be equal to n , and so $\phi_m = \phi_n$ also. The exact temporal autocorrelation of the model signal is therefore given by

$$\tilde{R}(i, j) = \frac{1}{2} \sum_{n=1}^N a_n^2 \cos(\omega_n|i-j|) + \Sigma(i, j). \quad (4.3)$$

The temporal autocorrelation is therefore a Toeplitz matrix, despite the fact that the underlying signals are nonstationary; as mentioned above, it is the same as the autocorrelation of the stationary semi-deterministic signal model.

Autocorrelation Matrix Estimates

There are several techniques for estimating the autocorrelation matrix from a data record; three of the most widely used methods are described below: the covariance method, which produces a symmetric positive definite estimate $\hat{\mathbf{R}}_C$ of the temporal autocorrelation $\tilde{\mathbf{R}}$, the biased Toeplitz estimate, which produces a symmetric positive definite Toeplitz estimate $\hat{\mathbf{R}}_B$, and the unbiased Toeplitz estimate, which produces a symmetric Toeplitz estimate $\hat{\mathbf{R}}_U$. Although the Toeplitz estimators are described as “biased” or “unbiased,” these descriptions apply only when they are used to estimate the autocorrelation of a stationary Gaussian signal. When the signal consists of deterministic sinusoids in noise, all of these estimators, including the covariance method, are biased estimates of both the temporal autocorrelation $\tilde{\mathbf{R}}$ and the statistical autocorrelation \mathbf{R} , for finite data records.

In the following sections, a brief description of each of these estimators will be presented. The terminology established in previous chapters will be retained; thus, the number of samples in the data record $y[k]$ will be denoted by L , the dimension of the autocorrelation estimate by M , and the actual number of (complex) sinusoidal signals by $N = 2N_R$.

The Covariance Estimate

The most widely used method of autocorrelation matrix estimation for subspace techniques is the so-called “covariance” method. Despite its name, the covariance method produces an estimation of the autocorrelation, not the covariance. The

elements of the covariance estimate $\hat{\mathbf{R}}_C$ are given by

$$\hat{R}_C(i, j) = \frac{1}{L - M + 1} \sum_{k=0}^{L-|i-j|} y[i+k]y[j+k]. \quad (4.4)$$

A convenient representation of $\hat{\mathbf{R}}_C$ is as the product of the $L - M + 1 \times M$ Toeplitz data matrix

$$\mathbf{Y} = \begin{bmatrix} y[M-1] & y[M-2] & y[M-3] & \dots & y[0] \\ y[M] & y[M-1] & y[M-2] & & y[1] \\ y[M+1] & y[M] & y[M-1] & & y[2] \\ \vdots & & & \ddots & \vdots \\ y[L-1] & y[L-2] & y[L-3] & \dots & y[L-M] \end{bmatrix} \quad (4.5)$$

and its transpose, that is,

$$\hat{\mathbf{R}}_C = \mathbf{Y}^T \mathbf{Y}. \quad (4.6)$$

From this representation, it is clear that $\hat{\mathbf{R}}_C$ is positive semidefinite.

By comparing the definition of $\hat{\mathbf{R}}_C$ in equation (4.4) with the temporal autocorrelation $\tilde{\mathbf{R}}$ of (4.1), we can see that for any finite M , the limit of $\hat{\mathbf{R}}_C$ as L goes to infinity is $\tilde{\mathbf{R}}$. Expanding $\hat{\mathbf{R}}_C$ as

$$\begin{aligned} \hat{R}_C(i, j) &= \frac{1}{L - M + 1} \left\{ \sum_{k=0}^{L-|i-j|} x[i+k]x[j+k] \right. \\ &\quad + \sum_{k=0}^{L-|i-j|} n[i+k]x[j+k] \\ &\quad + \sum_{k=0}^{L-|i-j|} x[i+k]n[j+k] \\ &\quad \left. + \sum_{k=0}^{L-|i-j|} n[i+k]n[j+k] \right\}, \end{aligned}$$

and comparing each term with the corresponding term of equation (4.2), we can also see that there are two distinct sources of error in the covariance estimate. First, there is an error that results from the finite sample length, because the first summation term of the estimate is not equal to the temporal autocorrelation of the noise-free signal $x[k]$:

$$\begin{aligned}
 & \frac{1}{L - M + 1} \sum_{k=0}^{L-|i-j|} x[i+k]x[j+k] \\
 &= \frac{1}{L - M + 1} \sum_{m=1}^N \sum_{n=1}^N a_m a_n \sum_{k=0}^{L-|i-j|} \cos(\omega_m(i+k) + \phi_m) \cos(\omega_n(j+k) + \phi_n) \\
 &\neq \frac{1}{2} \sum_{n=1}^N a_n^2 \cos(\omega_n|i-j|).
 \end{aligned}$$

This error occurs even when there is no noise; it is due solely to the use of a finite sample in calculating $\hat{\mathbf{R}}_C$.

The remaining three terms in the expansion are nonzero only in the presence of noise. The two cross terms are weighted sums of zero-mean Gaussian random variables, and so are normally distributed with a variance that decreases linearly with the number of samples. If $\Sigma = \mathbf{I}$, the mean of the cross terms is zero, but this is not necessarily the case for colored noise. The estimated noise autocorrelation $\hat{\Sigma}$ is a matrix random variable that is Wishart distributed with expected value Σ [39, p. 249]. The magnitude of the error in $\hat{\mathbf{R}}_C$ due to these three terms is dependent on both the number of samples and the magnitude of Σ .

The Biased Toeplitz Estimate

The biased Toeplitz estimate is defined by the equation

$$\hat{R}_B(i, j) = \frac{1}{L} \sum_{k=0}^{L-|i-j|-1} y[k]y[k + |i - j|]. \quad (4.7)$$

Like the covariance estimate, the biased Toeplitz estimate may be written as a product of a Toeplitz matrix with its transpose, $\hat{\mathbf{R}}_B = \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}}$, where

$$\tilde{\mathbf{Y}} = \begin{bmatrix} y[0] & 0 & 0 & \dots & 0 \\ y[1] & y[0] & 0 & & 0 \\ y[2] & y[1] & y[0] & & 0 \\ \vdots & & & \ddots & \vdots \\ y[M-1] & y[M-2] & y[M-3] & & y[0] \\ \vdots & & & \ddots & \vdots \\ y[L-1] & y[L-2] & y[L-3] & & y[L-M] \\ 0 & y[L-1] & y[L-2] & & y[L-M+1] \\ 0 & 0 & y[L-1] & & y[L-M+2] \\ 0 & 0 & 0 & & y[L-M+3] \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & y[L-1] \end{bmatrix}.$$

The biased Toeplitz estimate is closely related to the covariance estimate, because

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \mathbf{L} \\ \mathbf{Y} \\ \mathbf{U} \end{bmatrix},$$

where \mathbf{L} and \mathbf{U} are $M \times M$ lower and upper triangular matrices, respectively. Because the biased Toeplitz estimate is the same as as the covariance estimate of a data sequence that has been extended with M zeros at the start and end, it is also sometimes called the pre- and post-windowed estimate. Since it is an outer product, it is always positive semidefinite. As the number of samples increases, $\hat{\mathbf{R}}_B$ approaches $\hat{\mathbf{R}}_C$,

because the contributions of \mathbf{L} and \mathbf{U} to the product $\hat{\mathbf{R}}_B = \tilde{\mathbf{Y}}^T \tilde{\mathbf{Y}}$ become negligible; its limit as L goes to infinity is therefore $\tilde{\mathbf{R}}$, the same as the covariance estimate.

The Unbiased Toeplitz Estimate

The unbiased Toeplitz estimate is identical to the biased Toeplitz estimate, except that the scaling of the estimate as a function of the lag $|i - j|$ has been modified so that, for a stationary Gaussian process, the estimate is unbiased. The entries of the autocorrelation matrix estimate are given by

$$\hat{R}_U(i, j) = \frac{1}{L - |i - j|} \sum_{k=0}^{L-|i-j|-1} y[k]y[k + |i - j|]. \quad (4.8)$$

For a fixed M , the unbiased estimate also converges to the temporal autocorrelation as L goes to infinity, because the effect of the change in scaling approaches zero. Unlike the previous two estimators, however, the unbiased Toeplitz estimate is not necessarily positive definite for finite data records.

Comparison of Autocorrelation Estimators

In the sections above, we have shown that the subspace methods with a Toeplitz estimate of the autocorrelation matrix have the same asymptotic performance as the versions that use a covariance estimate, because each of these estimates approaches the same limit as the data record length L goes to infinity. Still left unanswered is the question of how well a subspace method will perform using Toeplitz estimates derived from finite data records. The first step in answering this question is to compare the performance of the autocorrelation estimators as a function of matrix dimension, data record length, and signal-to-noise ratio.

To provide a common test case for all three estimators, a set of 1000 data records was generated; each data record contained samples of a process made up

of 2 sinusoidal signals ($a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 2.01062$, $\phi_2 = -0.4$), in additive white Gaussian noise of variance $\sigma^2 = 0.01$. For each record, autocorrelation matrix estimates of dimensions $5 \leq M \leq 195$ were generated using each of the three techniques described above. The difference (in the 2-norm) between each estimate and the exact temporal autocorrelation, $\|\hat{\mathbf{R}} - \bar{\mathbf{R}}\|_2$, was then calculated for each of the estimates.

The first set of estimates was for a data record length of $L = 200$. The results are shown in Figures 4.1 and 4.2; the unbiased Toeplitz estimator $\hat{\mathbf{R}}_U$ is consistently the most accurate of the three estimators, and the biased Toeplitz estimator $\hat{\mathbf{R}}_B$ is more accurate than the covariance estimator $\hat{\mathbf{R}}_C$ when the size of the estimated autocorrelation matrix is small. The covariance estimate approaches the accuracy of the unbiased Toeplitz estimate for intermediate values of M , before the effects of the small data record length reduce its accuracy again for $M \approx L$.

When the size of the autocorrelation matrix approaches the number of points in the data record (in this case, 200), estimation of the autocorrelation at large lags is performed using only a small number of samples. This causes large errors in matrix elements far from the main diagonal. The covariance estimator is particularly sensitive to this effect, as indicated by the dramatic increase in error as M approaches 200. The biased Toeplitz estimator suffers much less from this problem, and the unbiased Toeplitz estimator shows only a slight increase for $M = 195$, the largest matrix considered. The increasing error in the biased Toeplitz estimator for large M is due to bias in its estimates of large lags, not to an increase in their variance.

When a larger number of samples are available, the differences between the three estimators are less pronounced. Figures 4.3 and 4.4 show the accuracy of the estimators for the same signals and noise power, but for a data record length of $L = 2000$. Again, the unbiased estimator is uniformly more accurate over the entire

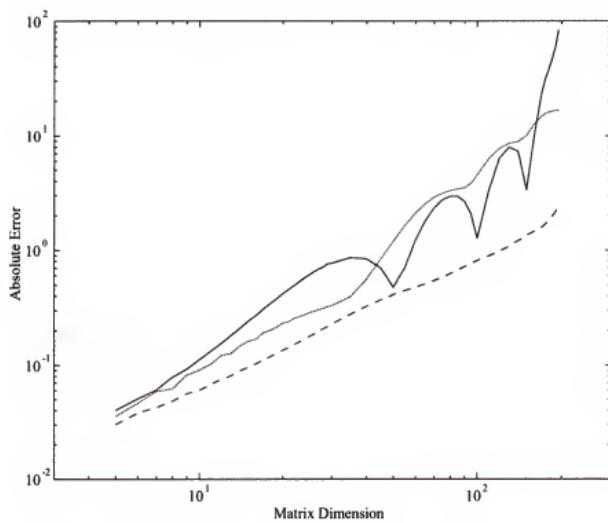


Figure 4.1: Mean Absolute Error $\|\bar{\mathbf{R}} - \hat{\mathbf{R}}\|_2$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for $L = 200$

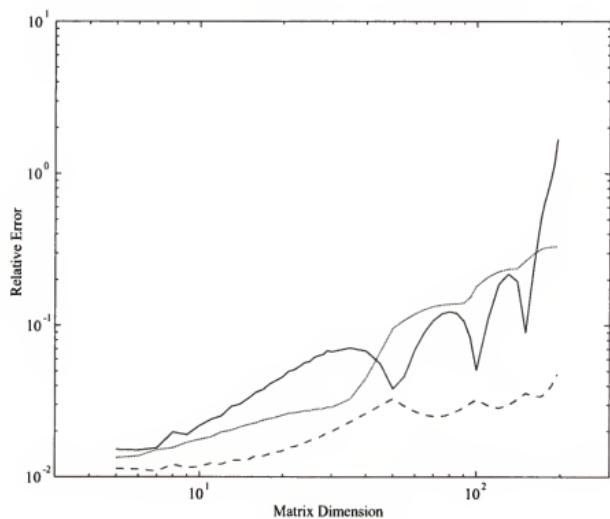


Figure 4.2: Mean Relative Error $\|\tilde{\mathbf{R}} - \hat{\mathbf{R}}\|_2 / \|\tilde{\mathbf{R}}\|_2$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for $L = 200$

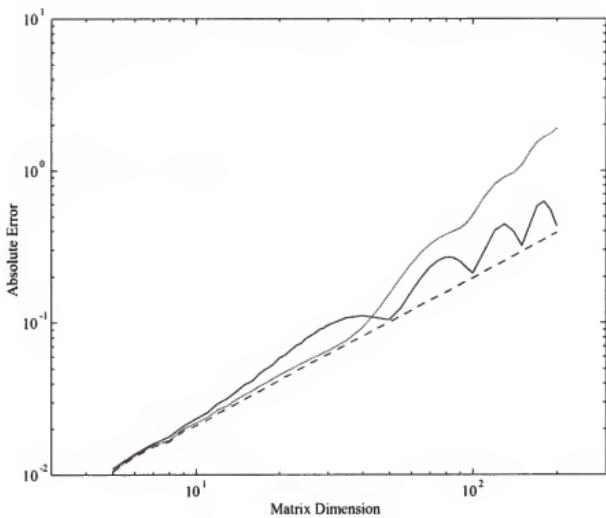


Figure 4.3: Mean Absolute Error $\|\tilde{\mathbf{R}} - \hat{\mathbf{R}}\|_2$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for $L = 2000$

range of M . As the theoretical analysis in an earlier section predicted, each of the estimators has become more accurate as more data are used in its computation.

This comparison has been presented for a single example, but similar numerical experiments have been conducted for a wide range of signals, and the conclusions presented here remain valid. These experiments indicate that the unbiased Toeplitz estimator is the most accurate of the three methods considered, in terms of the error in the 2-norm, when estimating autocorrelation matrices of signals consistent with the deterministic signal model.

Although the unbiased Toeplitz estimator is a superior estimator of the autocorrelation matrix, the subspace methods require estimates of invariant subspaces, and it is not necessarily true that a more accurate estimate of the autocorrelation matrix produces a more accurate estimate of its invariant subspaces. In the following section, we will examine the relation between the accuracy of a matrix estimate and the accuracy of subspace estimates.

Effects of Errors in Autocorrelation Estimation

We have seen that the various estimators of the autocorrelation matrix have differing error magnitudes under different conditions. In order to assess the impact of these errors, it is necessary to determine the effect of a perturbation of the autocorrelation matrix on its invariant subspaces. Since the temporal autocorrelation of the deterministic signal model is Toeplitz, and since the subspace techniques produce exact frequency estimates when the temporal autocorrelation is used as input, an estimate that produces a Toeplitz matrix close to the temporal autocorrelation will result in good frequency estimates.

It is possible to bound the effects of errors in estimating the autocorrelation on the accuracy of the estimated subspaces; for this purpose, a distance measure between subspaces is necessary. It is important to recognize that we do not want to measure

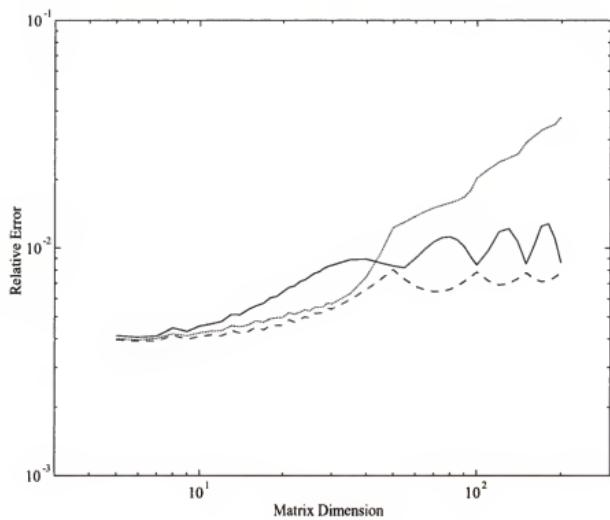


Figure 4.4: Mean Relative Error $\|\tilde{\mathbf{R}} - \hat{\mathbf{R}}\|_2 / \|\tilde{\mathbf{R}}\|_2$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for $L = 2000$

the distance between matrices that span the subspaces. For example, the matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}$$

all span the same subspace, even though the matrix norm of the difference between any two of them is nonzero. A proper measure of the distance between two l -dimensional subspaces of n -dimensional space is provided by the k canonical angles $\pi/2 \geq \Theta_1 \geq \Theta_2 \geq \dots \geq \Theta_k \geq 0$ between the subspaces, where $k = l$ if $l \leq n/2$, and $k = n - l$ if $l > n/2$ [40, 41]. Often, the $n \times n$ diagonal matrix of canonical angles $\Theta = \text{diag}(\Theta_1, \Theta_2, \dots, \Theta_k, 0, \dots, 0)$ is used to represent the difference between two subspaces. The canonical angles are zero if and only if the two subspaces are identical, so they can be used to identify identical subspaces. In addition, the sines of the canonical angles may be used as distance measures between subspaces. In particular, $\sin \Theta_1(\mathcal{X}, \mathcal{Y})$, the sine of the largest canonical angle between two subspaces \mathcal{X} and \mathcal{Y} , possesses all the properties required of a distance: $\sin \Theta_1(\mathcal{X}, \mathcal{Y}) = 0$ if and only if $\mathcal{X} \equiv \mathcal{Y}$, $\sin \Theta_1(\mathcal{X}, \mathcal{Y}) = \sin \Theta_1(\mathcal{Y}, \mathcal{X})$, and $\sin \Theta_1(\mathcal{X}, \mathcal{Y}) = D_1$, $\sin \Theta_1(\mathcal{Y}, \mathcal{Z}) = D_2$ implies that $\sin \Theta_1(\mathcal{X}, \mathcal{Z}) \leq D_1 + D_2$ (the triangle inequality).

With a distance measure selected, we may proceed to examine the effects of errors in the autocorrelation matrix estimate on the invariant subspaces of the matrix. The following two theorems, known as the Sin Θ and Sin 2Θ theorems, are due to Davis and Kahan [40]. The forms of the two theorems given here have been translated into the terminology used in Chapter 3, and specialized to deal only with the norm of the perturbation. The original work contains far more than the small fragment presented here; more complete forms of these theorems will be presented in Chapter 6, where they will serve as the basis for error detection in an eigendecomposition algorithm.

THEOREM 4.1 (SIN Θ THEOREM – PERTURBATION FORM) *Let $\tilde{\mathbf{R}}$ be the $M \times M$ temporal autocorrelation matrix of a real signal composed of $N_R = N/2$ deterministic sinusoids of powers $P_1 \geq P_2 \geq \dots \geq P_{N_R} > 0$, in additive white Gaussian noise of variance σ^2 . Since $\tilde{\mathbf{R}}$ must be positive semidefinite, its eigendecomposition may be expressed as*

$$\tilde{\mathbf{R}} = [\mathbf{E}_S, \mathbf{E}_N]^T \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_N, \sigma^2, \dots, \sigma^2) [\mathbf{E}_S, \mathbf{E}_N],$$

where the columns of the $M \times N$ matrix \mathbf{E}_S span the exact signal subspace \mathcal{S} , and the columns of the $M \times N - M$ matrix \mathbf{E}_N span the exact noise subspace \mathcal{N} . (Because \mathbf{R} is the exact autocorrelation, $\lambda_{2k-1} = \lambda_{2k} = P_k + \sigma^2$ for $1 \leq k \leq N_R$.)

In addition, let $\hat{\mathbf{R}} = \tilde{\mathbf{R}} + \mathbf{H}$ be a symmetric approximation to the exact autocorrelation matrix, and express the eigendecomposition of $\hat{\mathbf{R}}$ by

$$\hat{\mathbf{R}} = [\hat{\mathbf{E}}_S, \hat{\mathbf{E}}_N]^T \operatorname{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_N, \hat{\lambda}_{N+1}, \dots, \hat{\lambda}_M) [\hat{\mathbf{E}}_S, \hat{\mathbf{E}}_N],$$

where $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_M$, the N columns of $\hat{\mathbf{E}}_S$ span the approximate signal subspace $\hat{\mathcal{S}}$, and the $N - M$ columns of \mathbf{E}_N span the approximate noise subspace $\hat{\mathcal{N}}$.

Then:

$$\|\sin \Theta(\mathcal{S}, \hat{\mathcal{S}})\|_2 = \sin \Theta_1(\mathcal{S}, \hat{\mathcal{S}}) \leq \frac{\|\mathbf{H}\|_2}{\hat{\lambda}_N - \sigma^2},$$

and

$$\|\sin \Theta(\mathcal{N}, \hat{\mathcal{N}})\|_2 = \sin \Theta_1(\mathcal{N}, \hat{\mathcal{N}}) \leq \frac{\|\mathbf{H}\|_2}{P_{N_R} + \sigma^2 - \hat{\lambda}_{N+1}}.$$

The Sin Θ theorem illustrates an important property of invariant subspaces: the effect of a perturbation, in this case \mathbf{H} , depends not only on the magnitude of the perturbation, but also on the amount of separation between the eigenvalues associated with the exact invariant subspace and the other eigenvalues of the matrix.

An intuitive insight into this connection can be developed by considering the problem in signal processing terms. If the power in the weakest signal P_{N_R} , is much larger than the power in the noise σ^2 , then we would expect the performance of a frequency estimation technique to be relatively insensitive to small errors in estimating the autocorrelation. On the other hand, if the power in the smallest signal is comparable to the noise power, a small error in estimating the autocorrelation can result in the smallest signal being mistaken for noise, and a component of the noise being taken for a signal.

The Sin Θ theorem exhibits these effects in perturbation bounds that are ratios of the size of the error, $\|\mathbf{H}\|_2$, to the separation between the eigenvalues associated with one of the exact subspaces and those associated with the complementary estimated subspace. It is reasonable to assume that, if the error is small, the difference between the eigenvalues of the exact and estimated subspaces will be small also. This assumption is confirmed by the following theorem [41, p. 203]:

THEOREM 4.2 *Let $\tilde{\mathbf{R}}$, $\hat{\mathbf{R}}$, and \mathbf{H} and their eigenvalues be as defined in Theorem 4.1. Then, for any unitarily invariant norm,*

$$\|\text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_M) - \text{diag}(\lambda_1, \dots, \lambda_M)\| \leq \|\mathbf{H}\|.$$

In particular, for the 2-norm,

$$\|\text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_M) - \text{diag}(\lambda_1, \dots, \lambda_M)\|_2 = \max_i |\hat{\lambda}_i - \lambda_i| \leq \|\mathbf{H}\|_2.$$

From this theorem and the Sin Θ theorem, we can see that the sensitivity of an invariant subspace of a symmetric matrix to perturbations is determined by the smallest distance between any eigenvalue associated with that subspace and any eigenvalue associated with the complementary subspace. In the context of subspace

signal processing methods, this means that the accuracy to which the signal or noise subspace may be determined from an estimate of the autocorrelation matrix depends both on the accuracy of the autocorrelation estimate and on the signal-to-noise ratio of the weakest signal.

The Sin Θ theorem assumes that the eigenvalues of both the exact temporal autocorrelation matrix $\tilde{\mathbf{R}}$ and the estimate $\hat{\mathbf{R}}$ are known. In many cases, only the eigenvalues of $\hat{\mathbf{R}}$ are available. In this case, a somewhat weaker bound is provided by the Sin 2Θ theorem:

THEOREM 4.3 (SIN 2Θ THEOREM – PERTURBATION FORM) *Let the matrices $\tilde{\mathbf{R}}$, $\hat{\mathbf{R}}$, and \mathbf{H} , their eigendecompositions, and their invariant subspaces be as defined in Theorem 4.1. Then*

$$\|\sin 2\Theta(\mathcal{S}, \hat{\mathcal{S}})\|_2 = \sin 2\Theta_1(\mathcal{S}, \hat{\mathcal{S}}) \leq \frac{2\|\mathbf{H}\|_2}{\hat{\lambda}_N - \hat{\lambda}_{N+1}},$$

and

$$\|\sin 2\Theta(\mathcal{N}, \hat{\mathcal{N}})\|_2 = \sin 2\Theta_1(\mathcal{N}, \hat{\mathcal{N}}) \leq \frac{2\|\mathbf{H}\|_2}{\hat{\lambda}_N - \hat{\lambda}_{N+1}}.$$

As $\|\mathbf{H}\|_2$ approaches zero, the Sin Θ and Sin 2Θ theorems are asymptotically equivalent, since $\sin \Theta_1$ approaches $\frac{1}{2} \sin 2\Theta_1$ as Θ_1 approaches zero. When the perturbation is large, however, the bound given by the Sin 2Θ theorem is not as strong, since a small $\sin 2\Theta_1$ can also occur for Θ_1 near $\pi/2$. This situation occurs because, without knowledge of the exact eigenvalues, it is impossible to be certain that the eigenvectors chosen to span the approximate subspace are associated with the correct eigenvalues. How big a perturbation is required before there is a danger of mistaking an eigenvalue associated with the noise subspace for one associated with the signal subspace, or vice versa? An answer is provided by the following theorem [40, p. 37]:

THEOREM 4.4 *Let $\tilde{\mathbf{R}}$ and $\hat{\mathbf{R}}$, their eigendecompositions, and their invariant subspaces be as defined in Theorem 4.1. If $\|\mathbf{H}\|_2 = \|\tilde{\mathbf{R}} - \hat{\mathbf{R}}\|_2 < P_{N_R}/2$, then $\Theta_1 < \pi/4$.*

In essence, this theorem shows that if the error in the matrix estimate is small, it is impossible for an eigenvector belonging to one subspace to be mistaken for another; by comparing this bound with Theorem 4.2, we can see that the condition $\|\mathbf{H}\|_2 < P_{N_R}/2$ guarantees that $\lambda_N > \hat{\lambda}_{N+1}$ and $\hat{\lambda}_N > \lambda_{N+1}$, so that the eigenvalues that determine membership in the two subspaces remain in the correct order; this means that the eigenvectors that define the subspaces can still be properly identified from the perturbed eigenvalues.

Unfortunately, bounds involving only the error $\|\mathbf{H}\|$ are relatively blunt instruments; there are many situations where the error in the subspace estimate is far less than the upper limit set by applying the Sin Θ and Sin 2Θ theorems to $\|\mathbf{H}\|$. For example, \mathbf{A} and $\mathbf{A} + \alpha\mathbf{I}$ have identical invariant subspaces, but the norm of their difference is $|\alpha|$, which can be arbitrarily large. The bound provided by the Sin Θ and Sin 2Θ theorems applies in one direction only: although the invariant subspaces of two matrices must be close if the norm of their difference is small, a large difference does not imply that the invariant subspaces are far apart. This means that, knowing $\|\mathbf{H}\|$, we can bound the error in the subspace estimates; we cannot, however, assert that a better estimate of $\tilde{\mathbf{R}}$ must necessarily lead to a better estimate of the subspace.

Comparison of Frequency Estimates

In this section, the performance of the three autocorrelation matrix estimators is compared on the basis of their accuracy in frequency estimation. Several numerical experiments were conducted using the root-MUSIC and TLS-ESPRIT methods for frequency estimation; although the results shown here were computed using TLS-ESPRIT, the results for MUSIC are very similar, and lead to essentially the same

conclusions. In the following chapters, fast methods will be described for performing frequency estimation using Toeplitz estimates of the autocorrelation matrix; for these examples, however, the same standard $\mathcal{O}(M^3)$ computational techniques were used for both Toeplitz and non-Toeplitz matrices.

From our analysis of autocorrelation estimation techniques, we would expect the performance penalty for the use of Toeplitz autocorrelation estimators to be small when there are many samples in the input data record, since the Toeplitz and covariance estimators approach the same limit for large data records. In addition, we would also expect that the differences between the performance of the different estimators would decrease as the signal-to-noise ratio decreases, because the contribution of the Toeplitz noise autocorrelation Σ to the overall estimate becomes larger. We will see below that these expectations are borne out in practice.

The signals and noise are the same as used earlier to compare the accuracy of the autocorrelation matrix estimates ($a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 2.01062$, $\phi_2 = -0.4$, $\sigma^2 = 0.01$). In this section we will examine the bias and variance in the frequency estimate $\hat{\omega}_1$ when the various autocorrelation matrix estimates are used. The mean squared error in the estimate $\hat{\omega}_1$ is shown in Figure 4.5. (Although the Bhattacharyya bound limits the variance of an estimate, not the mean squared error, it is also shown to provide an indication of the quality of the estimate.)

The mean squared error in the frequency estimate for each of the three matrix estimators shows important differences from the norm of the error in $\hat{\mathbf{R}}$ shown in Figure 4.2. For small autocorrelation matrices, the error in all three frequency estimates decreases rapidly with the size of the autocorrelation matrix, even though the relative error of each matrix estimate $\|\hat{\mathbf{R}} - \tilde{\mathbf{R}}\|_2$ is increasing slowly. Each of the estimates shown uses the entire input data record to calculate $\hat{\mathbf{R}}$, so the Cramér-Rao and Bhattacharyya bounds are the same for any M . However, a qualitative argument

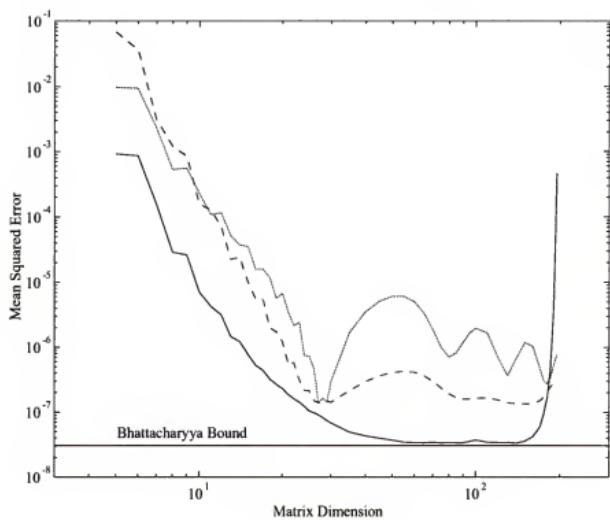


Figure 4.5: Mean Squared Error in $\hat{\omega}_1$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for ESPRIT with $L = 200$

based on the statistical bounds provides a justification for the rapid improvement of the quality of the estimate as the size of $\hat{\mathbf{R}}$ increases.

In Chapter 2, we remarked that the Cramér-Rao bound for the variance of an estimator given L consecutive (coherent) samples of a signal decreased asymptotically as $1/L^3$, while the bound for an estimator given the same number of samples in K separate data records of length $L_{sep} = L/K$ (the multiple snapshot case in array processing), decreased asymptotically as $1/KL_{sep}^3$. An autocorrelation matrix of dimension M contains information on lags from $-(M-1)$ to $M-1$, and therefore implicitly “windows” the input data; if this window is much smaller than length of the data record, the benefit of the long coherent sampling interval is lost, and the estimate may be much worse than the optimum given by the statistical bounds.

From Figure 4.5, we can see that an autocorrelation matrix estimate whose dimensions are roughly $1/4$ of the data record length ($M \geq L/4$) is required before the variance of the estimate approaches the statistical bounds. Because the computational cost of a subspace estimate using conventional eigendecomposition techniques is $\mathcal{O}(M^3)$, this implies that approaching the statistical bounds with conventional methods requires a very large amount of computation for long data records.

The bias in each of the frequency estimates is shown in Figure 4.6. Most of the error using the Toeplitz estimates of $\tilde{\mathbf{R}}$ is due to bias; the bias reaches a plateau when the dimension of the autocorrelation matrix is approximately $1/8$ of the length of the input data record, and remains relatively constant thereafter. For all but very small autocorrelation matrices, the unbiased Toeplitz matrix produces a smaller bias in the frequency estimate than the biased Toeplitz matrix. The notch that appears at $M = 27$ is a coherent effect whose location depends on the frequency and phase of the signals; although it appears consistently, its exact location is difficult to predict, so it would be difficult to select M to take advantage of this notch. The estimates

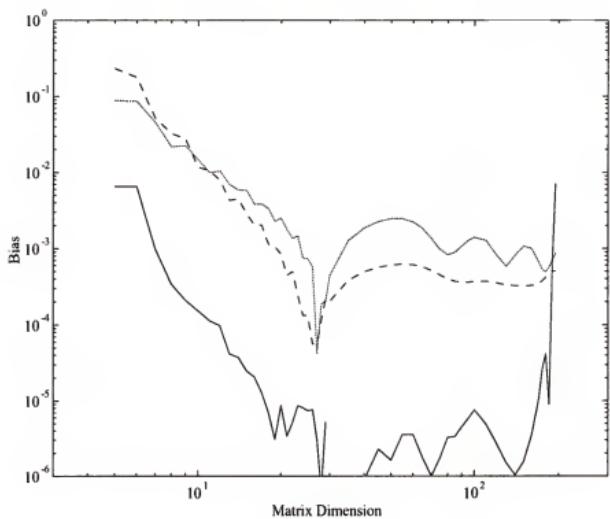


Figure 4.6: Bias of $\hat{\omega}_1$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for ESPRIT with $L = 200$

using the covariance method are effectively unbiased once a certain threshold M is reached; this threshold is approximately $1/10$ of the length of the data record.

The variance of the estimates using each of the three autocorrelation matrix estimates is shown in Figure 4.7. The covariance estimate approaches the Bhattacharyya bound once $M > L/4$, and remains close to the bound until finite data record effects increase the error for $M \approx L$. The variance for the Toeplitz estimates is only slightly greater than the variance using the covariance estimates, and is much less affected by the finite data record when $M \approx L$. In fact, both of the Toeplitz estimates slightly surpass the Bhattacharyya bound for $M \approx L$; this is possible because the bound applies to unbiased estimators, and both the Toeplitz-based estimators are biased.

A final aspect of the performance of these estimators is their resolution. The test case used above has $\Delta f = 0.02$, while $L = 200$, so that the frequency separation is a factor of four larger than the classical resolution limit $1/L$. Since the major advantage of the subspace techniques over simpler classical approaches is their high resolution, it is important to determine the effects of using Toeplitz autocorrelation estimates on the resolution. Figure 4.8 shows the mean squared error for each of the three estimators, with an input signal whose frequencies are separated by $\Delta f = 0.001$, which is a factor of five smaller than the classical resolution limit. The other signal parameters are identical to the previous cases ($a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 1.89124$, $\phi_2 = -0.4$, $\sigma^2 = 0.01$). As before, the Bhattacharyya bound is shown for reference.

The covariance estimator, as expected, resolves the signals once the autocorrelation matrix size exceeds $L/4$. The biased estimator also resolves the signals, but with a large bias, as can be seen from Figure 4.9, which shows the variance. Note that both of the Toeplitz estimates, which are biased, exceed the Bhattacharyya bound, while the covariance estimate does not approach the bound.

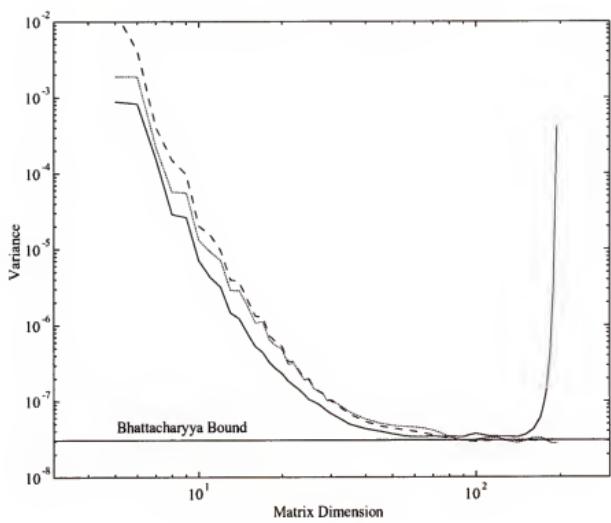


Figure 4.7: Variance of $\hat{\omega}_1$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for ESPRIT with $L = 200$

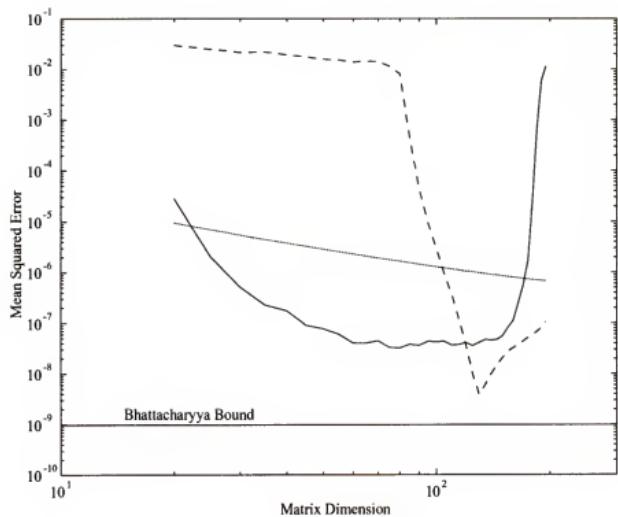


Figure 4.8: Mean Squared Error of $\hat{\omega}_1$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for ESPRIT with $\Delta f = 0.001$ and $L = 200$

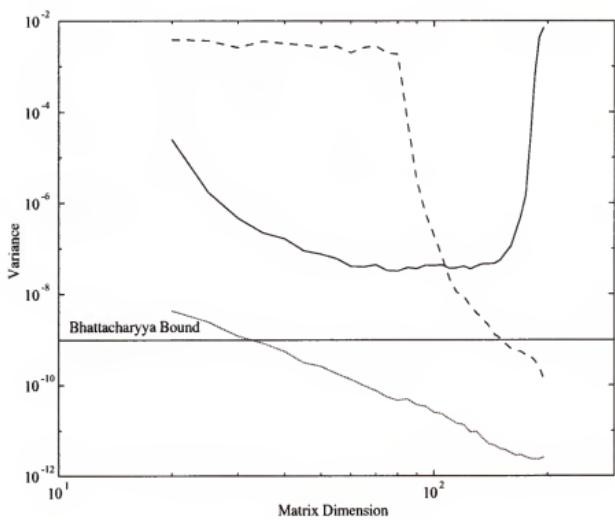


Figure 4.9: Variance of $\hat{\omega}_1$ for the Covariance (solid line), Biased (dotted line), and Unbiased (dashed line) Estimators for ESPRIT with $\Delta f = 0.001$ and $L = 200$

Although the biased estimator has the largest mean squared error, it has the lowest variance of any of the estimators. In fact, the biased estimator overestimates the separation of the two sinusoids by approximately a factor of two. This behavior is very sensitive to the phase and frequency of the two signals; in other similar cases, the biased estimator has failed to resolve the two signals. The unbiased estimator, by contrast, consistently exhibits the behavior shown: complete failure to resolve the signals until $M \approx L/2$, followed by an abrupt transition to high resolution.

It is clear from these results that the subspace estimators retain their high resolution when Toeplitz estimators of the autocorrelation matrix are used, provided that the estimators and the matrix size are properly chosen. Although the biased estimator can exhibit high resolution, as in the previous example, it also occasionally fails to resolve signals which are resolved with the covariance estimate. The unbiased estimator is more reliable, but when it is used, it is necessary that $M \geq L/2$, to ensure that high resolution is achieved. When these guidelines are followed, the mean squared error of the Toeplitz-based subspace techniques may be quite close to the error when the covariance estimate is used.

Based on the results shown above, we may conclude that, for a given data record length L and autocorrelation matrix dimension M , the covariance estimate $\hat{\mathbf{R}}_C$ produces the most accurate frequency estimates. In fact, if $M \geq L/4$, then estimates computed using $\hat{\mathbf{R}}_C$ are practically unbiased, and their variance approaches the minimum possible variance given by the Bhattacharyya bound. However, computing invariant subspaces when the covariance estimate is used requires $\mathcal{O}(M^3)$ floating-point operations, and since approaching the Bhattacharyya bound requires that M is at least $L/4$, this performance comes at a very high computational cost, especially for large data records.

By using a Toeplitz estimate of the autocorrelation matrix, we can dramatically reduce the computational cost of frequency estimation. In cases where the signal-to-noise ratio is moderate to low, or when there are many samples in the input data, the frequency estimation performance with a Toeplitz estimate can be very close to that obtained with a covariance estimate. Conversely, when the input data record is short and the signal-to-noise ratio high, the performance penalty will be greater. Although the accuracy of the estimates obtained will be somewhat less than if a covariance estimate of equal dimension were used, when the estimates are compared on the basis of equal computational cost, the estimate computed using a Toeplitz autocorrelation matrix may be equal or even superior to an estimate using a smaller covariance estimate of the autocorrelation matrix. A detailed comparison of accuracy and speed is presented in Chapter 9, where we will see that this is often the case.

CHAPTER 5

FAST ALGORITHMS FOR SOLVING TOEPLITZ EQUATIONS

Although solution of a general system of n linear equations requires $\mathcal{O}(n^3)$ floating-point operations, solution of a Toeplitz system may be performed much more efficiently. There are several fast algorithms for solving various types of Toeplitz systems, including the conventional “fast” $\mathcal{O}(n^2)$ methods such as the Levinson-Durbin and Trench algorithms [37], and a group of newer “superfast” methods, which are $\mathcal{O}(n \log^2 n)$ [42–45].

Because the fast eigendecomposition algorithms to be discussed in the following chapter rely for their efficiency on a fast method for solving Toeplitz equations, this chapter provides a brief review of the fast and superfast methods for solving Toeplitz systems. As in previous chapters, the algorithms are summarized using MATLAB-style notation.

Solving the Yule-Walker Equation

One of the simplest and most common Toeplitz systems is the Yule-Walker equation, a special system that arises in linear prediction problems. If a $n \times n$ Toeplitz matrix \mathbf{T}_n is formed from the autocorrelation of a stationary random process:

$$\mathbf{T}_n = \begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_1 & t_0 & & t_{n-2} \\ \vdots & & \ddots & \vdots \\ t_{n-1} & t_{n-2} & \dots & t_0 \end{bmatrix},$$

where t_i is the autocorrelation at lag i , then the optimum least-squares linear predictor for the process is given by the solution of the system

$$\begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_1 & t_0 & & t_{n-2} \\ \vdots & \ddots & \ddots & \vdots \\ t_{n-1} & t_{n-2} & \dots & t_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = - \begin{bmatrix} t_n \\ t_{n-1} \\ \vdots \\ t_1 \end{bmatrix}, \quad (5.1)$$

or

$$\mathbf{T}_n \mathbf{a} = -\mathbf{t}.$$

This system of equations is completely determined by $n + 1$ coefficients, $t_0 \dots t_n$. Toeplitz systems of the form given by equation (5.1) are called Yule-Walker equations, and are sometimes written in the alternate form

$$\begin{bmatrix} t_0 & t_1 & \dots & t_n \\ t_1 & t_0 & & t_{n-1} \\ \vdots & \ddots & \ddots & \vdots \\ t_n & t_{n-1} & \dots & t_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} E_n \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

or

$$\mathbf{T}_{n+1} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E_n \\ \mathbf{0} \end{bmatrix}$$

where E_n is the residual, or minimum prediction error. If \mathbf{T}_{n+1} is positive definite, the prediction errors for each of the Yule-Walker problems of increasing size defined by the leading principal submatrices of \mathbf{T}_{n+1} form a decreasing sequence $E_0 > E_1 > \dots > E_n > 0$; the zero-order residual E_0 is equal to t_0 .

The Levinson-Durbin Algorithm

The most widely used method for solution of the Yule-Walker equations is the Levinson-Durbin algorithm, which computes the solution by solving a nested series of Yule-Walker equations for $\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_n$, using the fact that the solution of a problem of size $n+1$ can be computed from the solution of a problem of size n . If $\mathbf{T}_n \mathbf{a}_n = -\mathbf{t}$, then

$$\begin{bmatrix} t_0 & \mathbf{t}^T \mathbf{F}^T \\ \mathbf{F} \mathbf{t} & \mathbf{T}_n \end{bmatrix} \mathbf{a}_{n+1} = -\begin{bmatrix} t_{n+1} \\ \mathbf{t} \end{bmatrix},$$

where \mathbf{F} is the “flip” matrix, with ones on its antidiagonal, and zeros elsewhere; multiplying a vector by \mathbf{F} reverses the order of its coefficients. The solution to the larger problem may be easily computed by using this relation, as shown in Algorithm 5.1 below. A complete discussion of the the Levinson-Durbin algorithm can be found in many sources; see, for example, Golub [37, p. 183].

ALGORITHM 5.1: LEVINSON-DURBIN ALGORITHM

```

function  $[a, e] = \text{levinson}(t, n)$ 
  (dimensions:  $a(1 : n), e(0 : n), t(0 : n)$ )
   $a(1) = -t(1)/t(0)$ 
   $\gamma_1 = a(1)$ 
   $\beta = t(0)$ 
   $e(0) = \beta$ 
  for  $k = 1 : n - 1$ 
     $\beta = (1 - \gamma_k^2) * \beta$ 
     $e(k) = \beta$ 
     $\gamma_{k+1} = -(\gamma_k + a(1 : k)^T * t(k : -1 : 1)) / \beta$ 
     $a(1 : k) = a(1 : k) + \gamma_{k+1} * a(k : -1 : 1)$ 
     $a(k + 1) = \gamma_{k+1}$ 
  end
   $e(n) = (1 - \gamma_n^2) * e(n - 1)$ 

```

Algorithm 5.1 computes the solution vector and the prediction errors at a cost of $2n^2 + 3n$ floating-point operations. The stability of the Levinson-Durbin algorithm

depends on the magnitude of all of the reflection coefficients γ_i being less than one; this is equivalent to requiring that all the prediction errors E_i be greater than zero, which will occur if and only if \mathbf{T} is positive definite.

A useful interpretation of the process of solving a Toeplitz system is provided by expressing it in terms of operations on polynomial matrices. The solution vector \mathbf{a} of the Yule-Walker problem defines a polynomial

$$A_k(z) = 1 + \sum_{i=1}^k a(i)z^i,$$

and a reversed polynomial

$$\bar{A}_k(z) = z^k A_k(z^{-1}) = z^k + \sum_{i=1}^k a(i)z^{k-i}.$$

In terms of these polynomials, the k th step of the Levinson-Durbin algorithm is given by

$$\begin{bmatrix} A_k(z) \\ \bar{A}_k(z) \end{bmatrix} = \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix} \begin{bmatrix} A_{k-1}(z) \\ \bar{A}_{k-1}(z) \end{bmatrix}.$$

The initial values $A_0(z)$ and $\bar{A}_0(z)$ are 1 and z^{-1} , respectively, so the solution to the Yule-Walker problem is given by

$$\begin{bmatrix} A_k(z) \\ \bar{A}_k(z) \end{bmatrix} = \left(\prod_{k=1}^n \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix} \right) \begin{bmatrix} 1 \\ z^{-1} \end{bmatrix}. \quad (5.2)$$

This way of expressing Toeplitz algorithms will be useful in illustrating the similarities between the Levinson-Durbin algorithm and the other approaches described in later sections.

The Yule-Walker Prediction Error

The Levinson-Durbin algorithm computes the vector $[E_0, E_1, \dots, E_n]$ of prediction errors for each of the Yule-Walker systems of size $0, 1, \dots, n$ that are solved in the course of computing the answer. This prediction error vector is important in eigendecomposition algorithms because it defines a diagonal matrix that is congruent to \mathbf{T}_{n+1} [46, 47]:

$$\mathbf{A}_{n+1}^T \mathbf{T}_{n+1} \mathbf{A}_{n+1} = \text{diag}(E_n, E_{n-1}, \dots, E_0),$$

where

$$\mathbf{A}_{n+1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ a_n(1) & 1 & & 0 & 0 \\ a_n(2) & a_{n-1}(1) & 0 & 0 \\ a_n(3) & a_{n-1}(2) & 0 & 0 \\ \vdots & & \ddots & \vdots \\ a_n(n) & a_{n-1}(n-1) & \dots & a_1(1) & 1 \end{bmatrix},$$

and \mathbf{a}_k is the solution to the Yule-Walker problem of size k .

Two important uses of the prediction error sequence are based on this congruence relation. First, the product of the prediction errors is the determinant of \mathbf{T}_{n+1} , because $\det(\mathbf{A}_{n+1}) = \det(\mathbf{A}_{n+1}^T) = 1$:

$$\begin{aligned} \det(\mathbf{T}_{n+1}) &= \det(\mathbf{A}_{n+1}^{-T} \text{diag}(E_n, \dots, E_0) \mathbf{A}_{n+1}^{-1}) \\ &= \frac{\det(\text{diag}(E_n, \dots, E_0))}{\det(\mathbf{A}_{n+1}^T) \det(\mathbf{A}_{n+1})} \\ &= \prod_{i=0}^n E_i, \end{aligned}$$

and so

$$E_n = \frac{\det(\mathbf{T}_{n+1})}{\det(\mathbf{T}_n)}.$$

The second use of the prediction errors is based on the Sylvester inertia theorem [37, p. 416], which states that, if $\mathbf{X} = \mathbf{P}^T \mathbf{Y} \mathbf{P}$, where \mathbf{X} and \mathbf{Y} are symmetric and \mathbf{P} is nonsingular, then \mathbf{X} and \mathbf{Y} have the same number of positive, negative, and zero eigenvalues. The matrix \mathbf{A}_{n+1} is unit lower triangular, so it is obviously nonsingular. Because the eigenvalues of a diagonal matrix are just the diagonal elements, the number of positive, negative, and zero eigenvalues of \mathbf{T}_{n+1} may be determined by simply counting the number of positive, negative, and zero E_i . Both these properties will be extremely useful in the fast eigendecomposition algorithms given in the next chapter.

The Schur Algorithm

An alternative method for solving the Yule-Walker equation is the Schur algorithm, which takes a somewhat different approach to the calculation of the solution. Algorithm 5.2 below is the Schur algorithm, which takes as input a pair of vectors $\alpha = t(1:n)$ and $\beta = t(0:n-1)$, and computes vectors \mathbf{u} and \mathbf{v} such that

$$\begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{u} \end{bmatrix} + \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix}. \quad (5.3)$$

Like the Levinson-Durbin algorithm, this algorithm also computes the sequence of prediction errors E_i . The properties and derivation of the Schur algorithm are discussed in much greater detail by Kailath [48, 49] and Therrien [50, p. 444].

ALGORITHM 5.2: SCHUR ALGORITHM

```

function  $[u, v, e] = \text{schur}(\alpha, \beta, n)$ 
  (dimensions:  $u(1 : n), v(1 : n), e(0 : n), \alpha(1 : n), \beta(1 : n)$ )
   $u(1 : n) = 0$ 
   $v(1) = 1$ 
   $v(2 : n) = 0$ 
   $\gamma_0 = 0$ 
  for  $k = 1 : n$ 
     $\tilde{\alpha}(k - 1) = -\alpha(k)$ 
     $\tilde{\beta}(k - 1) = \beta(k)$ 
     $\begin{bmatrix} \tilde{\alpha}(k - 2 : -1 : 0) \\ \tilde{\beta}(k - 1 : -1 : 1) \end{bmatrix} = \begin{bmatrix} 1 & -\gamma_{k-1} \\ -\gamma_{k-1} & 1 \end{bmatrix} \begin{bmatrix} \tilde{\alpha}(k - 1 : -1 : 1) \\ \tilde{\beta}(k - 1 : -1 : 1) \end{bmatrix}$ 
     $\gamma_k = \tilde{\alpha}(0) / \tilde{\beta}(0)$ 
     $\begin{bmatrix} u(1 : k) \\ v(1 : k) \end{bmatrix} = \begin{bmatrix} u(1 : k) \\ v(1 : k) \end{bmatrix} + \gamma_k * \begin{bmatrix} v(k : -1 : 1) \\ u(k : -1 : 1) \end{bmatrix}$ 
     $e(k - 1) = \tilde{\beta}(0)$ 
     $\tilde{\beta}(0) = (1 - \gamma_k^2) * \tilde{\beta}(0)$ 
  end
   $e(n) = \tilde{\beta}(0)$ 

```

Using the polynomials defined by

$$\begin{aligned}
 U_n(z) &= \sum_{i=1}^n u(i)z^i, \\
 V_n(z) &= \sum_{i=1}^n v(i)z^i, \\
 \bar{U}_n(z) &= z^n U_n(z^{-1}), \\
 \bar{V}_n(z) &= z^n V_n(z^{-1}),
 \end{aligned}$$

the result of the Schur algorithm may be expressed as

$$\begin{bmatrix} U_n(z) & V_n(z) \\ \bar{V}_n(z) & \bar{U}_n(z) \end{bmatrix} = \prod_{k=1}^n \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix}. \quad (5.4)$$

By combining equation (5.4) with the initial conditions given in equation (5.2), it is easily determined that the solution of the Yule-Walker equation is $A(z) = U(z) +$

$z^{-1}V(z)$, which is equivalent to equation (5.3). Solution of the Yule-Walker problem using the Schur algorithm is shown in Algorithm 5.3, which requires $4n^2 + n$ floating-point operations.

ALGORITHM 5.3: SCHUR YULE-WALKER SOLUTION

```

function  $[a, e] = \text{schuryw}(t, n)$ 
  (dimensions:  $a(1 : n), e(0 : n), t(0 : n)$ )
   $[u, v, e] = \text{schur}(t(1 : n), t(0 : n - 1), n)$ 
  for  $i = 1 : n - 1$ 
     $a(i) = u(i) + v(i + 1)$ 
  end
   $a(n) = u(n)$ 

```

Although the Schur algorithm is less efficient on a single processor than the Levinson-Durbin algorithm, it is better suited to parallel implementations, since no inner products are required in its computation. Much of the current interest springs from the fact that, using the Schur algorithm, solution of the Yule-Walker equations on a parallel machine with at least n processors is possible with an execution time of $\mathcal{O}(n)$ [51].

The “Superfast” Generalized Schur Algorithm

All of the algorithms discussed to this point are conventional fast algorithms, which are $\mathcal{O}(n^2)$. An efficient $\mathcal{O}(n \log^2 n)$ algorithm, based on the Schur algorithm above, was proposed by Ammar and Gragg [52, 53], and independently by de Hoog [44], and Musicus [54]. Although several $\mathcal{O}(n \log^2 n)$ methods for solving the Yule-Walker equation had been previously advanced [42, 43, 55], most of these methods were very inefficient, becoming faster than the conventional $\mathcal{O}(n^2)$ algorithms only for n greater than several thousand. In contrast, the generalized Schur algorithm¹ as

¹The term “generalized Schur” has also been used by Kailath and others to refer to the Schur algorithm described in the previous section. In this work, the term generalized Schur algorithm will be used only to refer to the $\mathcal{O}(n \log^2 n)$ algorithm described here; this usage is consistent with that of Ammar and Gragg.

implemented by Ammar and Gragg [45, 56], requires $8n \log_2^2 n + \mathcal{O}(n \log n)$ floating-point operations, making it less complex than the Levinson-Durbin algorithm for $n > 256$.

The key to the efficiency of the generalized Schur algorithm is a fast method for computing the Schur polynomials given by equation (5.4). By factoring the matrix produced by the Schur algorithm,

$$\begin{aligned} \begin{bmatrix} U_n(z) & V_n(z) \\ \bar{V}_n(z) & \bar{U}_n(z) \end{bmatrix} &= \prod_{k=1}^n \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix} \\ &= \prod_{k=n/2+1}^n \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix} \prod_{k=1}^{n/2} \begin{bmatrix} 1 & \gamma_k z \\ \gamma_k & z \end{bmatrix} \\ &= \begin{bmatrix} U'_{n/2}(z) & V'_{n/2}(z) \\ \bar{V}'_{n/2}(z) & \bar{U}'_{n/2}(z) \end{bmatrix} \begin{bmatrix} U_{n/2}(z) & V_{n/2}(z) \\ \bar{V}_{n/2}(z) & \bar{U}_{n/2}(z) \end{bmatrix}, \end{aligned}$$

the problem has been split in two; the first part is another Schur problem of half the size, and the second is the computation of U' , V' , \bar{U}' , and \bar{V}' .

It was shown by de Hoog that the computation of U' , V' , \bar{U}' , and \bar{V}' could also be transformed into a Schur problem of order $n/2$, whose input could be computed from the original input and the output of the first Schur problem, and that these computations required only polynomial multiplication and addition [44, 53]. The Schur problem of order n has thus been split into two Schur problems of order $n/2$.

If n can be factored as $n = 2^k N'$, the splitting can be continued recursively, until the problem reaches size N' , and the conventional Schur algorithm may be used from that point. Also, at a certain problem size N_T , the generalized Schur algorithm becomes less efficient than the conventional algorithm, and it is more effective to solve the smaller problems using the conventional algorithm; typically $N_T \approx 64$. In the de Hoog, Musicus, and Ammar and Gragg implementations, n was restricted to be a

power of two, but this is not necessary; this point is discussed in more detail below. The resulting procedure is shown in Algorithm 5.4, where the symbol “ \star ” indicates polynomial multiplication, or convolution.

ALGORITHM 5.4: GENERALIZED SCHUR ALGORITHM

```

function  $[u, v, e] = \text{gsa}(\alpha, \beta, n)$ 
  (dimensions:  $u(1 : n), v(1 : n), e(0 : n), \alpha(1 : n), \beta(1 : n)$ )
  if  $n < N_T$ 
     $[u, v, e] = \text{schur}(\alpha, \beta, n)$ 
  else
     $m = n/2$ 
     $[u_0, v_0, e(0 : m - 1)] = \text{gsa}(\alpha(1 : m), \beta(1 : m), m)$ 
     $\tilde{u}_0(1 : m + 1) = [0, u_0(m : -1 : 1)]$ 
     $\tilde{v}_0(1 : m + 1) = [0, v_0(m : -1 : 1)]$ 
     $z(1 : n) = \alpha \star v_0 - \beta \star u_0$ 
     $\alpha_1(1 : m) = z(m + 1 : n)$ 
     $z(1 : n) = \beta \star \tilde{v}_0 - \alpha \star \tilde{u}_0$ 
     $\beta_1(1 : m) = z(m + 1 : n)$ 
     $[u_1, v_1, e(m : n - 1)] = \text{gsa}(\alpha_1, \beta_1, m)$ 
     $u = u_1 \star \tilde{v}_0 + [u_0 \star v_1, 0]$ 
     $v = u_1 \star \tilde{u}_0 + [v_0 \star v_1, 0]$ 
  end
   $e(n) = (1 - u(n)^2)e(n - 1)$ 

```

Because polynomial multiplication can be performed using the FFT, the intermediate computations at each stage are $\mathcal{O}(n \log n)$, and the overall algorithm is $\mathcal{O}(n \log^2 n)$. Ammar and Gragg [45] have extensively optimized the algorithm, using split-radix FFT algorithms [57, 58] to perform the convolutions, and rearranging the computations to take full advantage of the results of earlier recursive steps. A detailed description of the algorithm and an annotated Fortran program that implements it are given in Appendix B.

Although the algorithm as described by de Hoog, Ammar, and Gragg is restricted to solving systems of size 2^k , it is important to note that the range of sizes can be greatly expanded by a simple modification. The algorithm recursively splits the problem until it has been divided into subproblems of size N_T or smaller; as long

as the size of the problem n can be factored as $n = 2^k N'$, where $N' \leq N_T$, then the generalized Schur algorithm can be applied. To achieve the highest possible efficiency, it is desirable to restrict N' to be a product of small primes, so that a prime factor FFT algorithm [58, 59] may be used to compute the convolutions. This modification allows the algorithm to be used for a much wider range of sizes, at the cost of a slight reduction in performance when n is a power of 2. Because it was designed to achieve the highest possible efficiency, the algorithm given in Appendix B uses a real-valued split-radix FFT, and so is restricted to $n = 2^k$, but a more flexible algorithm may be obtained by simply replacing the split-radix FFT with a real-valued prime factor FFT.

Performance Comparison

When the complexity of algorithms for Yule-Walker solution is measured in terms of floating-point operations, the Levinson-Durbin and generalized Schur algorithms have equal complexity for matrices of size $n = 256$. In practice, the better locality of reference of the Levinson-Durbin algorithm, compared to the FFTs used in the generalized Schur algorithm, gives the Levinson-Durbin algorithm a performance advantage that is not overcome until $n \approx 512$. The execution time for a single solution of the Yule-Walker equations on a general-purpose processor is shown in Figure 5.1.

The poor locality of reference of the FFT handicaps the generalized Schur algorithm somewhat on general-purpose processors. Microprocessors designed for digital signal processing, however, are optimized for maximum performance on the FFT. The results of profiling the generalized Schur algorithm indicate that it would be ideally suited for implementation on these processors, where its efficiency advantage over the Levinson-Durbin algorithm should be even greater than indicated by Figure 5.1.

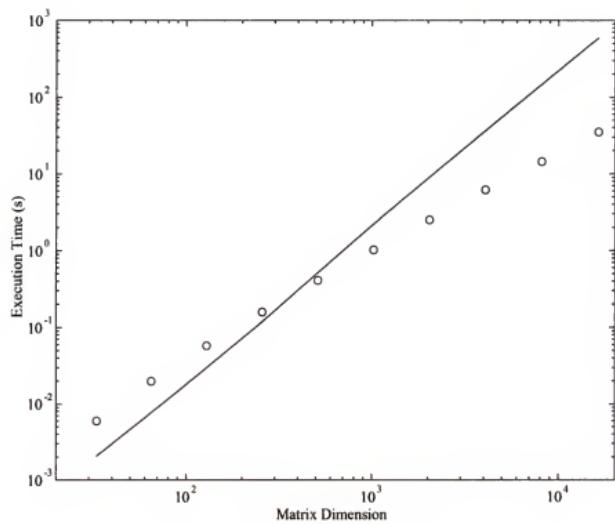


Figure 5.1: Execution Time versus Matrix Dimension for the Levinson-Durbin algorithm (solid line) and the generalized Schur algorithm (circles)

General Toeplitz Systems

All of the algorithms described to this point solve a very restricted type of Toeplitz system, the Yule-Walker equation. Although fast $\mathcal{O}(n^2)$ methods, such as Trench's algorithm [37], are available for solving general Toeplitz systems, in many cases the systems encountered in signal processing can be solved more efficiently by making use of two important properties of Toeplitz matrices.

Toeplitz Matrix-Vector Products

Because the multiplication of a vector by a Toeplitz matrix is equivalent to convolution, the product of a vector with a Toeplitz matrix may be efficiently computed using the fast Fourier transform [59, 60]. If the elements of the first row and column of a Toeplitz matrix \mathbf{T}_n are contained in the vectors t_{row} and t_{col} , the vector $\mathbf{y} = \mathbf{T}_n \mathbf{x}$ is computed by the following algorithm.

ALGORITHM 5.5: TOEPLITZ MATRIX-VECTOR MULTIPLICATION

```

function  $y = \text{tmvmul}(t_{row}, t_{col}, x, n)$ 
  (dimensions:  $y(1 : n)$ ,  $t_{row}(0 : n - 1)$ ,  $t_{col}(0 : n - 1)$ ,  $x(1 : n)$ )
  choose  $N \geq 2n + 1$ , a convenient length for the FFT
   $\tilde{t}(1 : N) = [0, \dots, 0, t_{row}(n - 1 : -1 : 1), t_{col}(0 : n - 1)]$ 
   $\tilde{x}(1 : N) = [x(1 : n), 0, \dots, 0]$ 
   $\tau = \text{fft}(\tilde{t}, N)$ 
   $\chi = \text{fft}(\tilde{x}, N)$ 
  for  $i = 1 : N$ 
     $\zeta(i) = \tau(i) * \chi(i)$ 
  end
   $\tilde{y} = \text{ifft}(\zeta, N)$ 
   $y = \tilde{y}(N - n + 1 : N)$ 

```

If \mathbf{T}_n and \mathbf{x} are real, the split-radix algorithm may be used to compute the forward and inverse FFTs, at a total cost of about $6N \log_2 N - 11N$ floating-point operations [57–59]. Matrix-matrix multiplication where one of the matrices is Toeplitz

may be performed even more efficiently, since the transform τ computed from \mathbf{T}_n need only be computed once.

The Gohberg-Semencul Relation

Using the solution of the Yule-Walker equation and the fast Toeplitz multiplication algorithm given above, it is often possible to solve a general Toeplitz system with only $\mathcal{O}(n \log n)$ additional floating-point operations [60], using a property of nonsingular Toeplitz matrices derived by Gohberg and Semencul [61].

THEOREM 5.1 (GOHBERG-SEMENCUL) *Let \mathbf{T}_{n+1} and its largest leading principal submatrix \mathbf{T}_n be nonsingular symmetric Toeplitz matrices, and \mathbf{a} be the solution to the Yule-Walker problem*

$$\mathbf{T}_{n+1} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E_n \\ \mathbf{0} \end{bmatrix},$$

with $E_n \neq 0$. Then \mathbf{T}_{n+1} is nonsingular, and its inverse is given by

$$\mathbf{T}_{n+1}^{-1} = \frac{1}{E_n} (\mathbf{L} \mathbf{L}^T - \mathbf{M} \mathbf{M}^T),$$

where

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a(1) & 1 & 0 & & 0 \\ a(2) & a(1) & 1 & & 0 \\ \vdots & & & \ddots & \vdots \\ a(n) & a(n-1) & a(n-2) & \dots & 1 \end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ a(n) & 0 & & 0 & 0 \\ a(n-1) & a(n) & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ a(1) & a(2) & \dots & a(n) & 0 \end{bmatrix}.$$

Because \mathbf{L} and \mathbf{M} are Toeplitz, their matrix-vector products may be computed efficiently using the FFT. This allows the general Toeplitz problem $\mathbf{T}_{n+1}\mathbf{x} = \mathbf{y}$, where \mathbf{T}_{n+1} satisfies the requirements of the Gohberg-Semencul relation, to be solved by computing

$$\mathbf{T}_{n+1}^{-1}\mathbf{y} = \frac{1}{E_n}(\mathbf{L}\mathbf{L}^T - \mathbf{M}\mathbf{M}^T)\mathbf{y},$$

using Algorithm 5.5 to compute the matrix-vector products. Thus, a general Toeplitz system may be solved in $\mathcal{O}(n \log n)$ operations once the solution to the Yule-Walker problem is known. Since the superfast algorithms can compute a Yule-Walker solution in $\mathcal{O}(n \log^2 n)$ operations, a general Toeplitz system can be solved without increasing the asymptotic complexity of the superfast algorithms.

The form of the Gohberg-Semencul relation given above has been specialized to the case of symmetric nonsingular Toeplitz matrices. Although this is sufficient for our needs, versions exist that are more general. The Gohberg-Semencul relation has a general form for nonsymmetric Toeplitz matrices [61], an extension of the relation to the case where \mathbf{T}_{n+1} may be singular has been given by Zhong [62], and the possibility of a more efficient solution for the singular case has been suggested by Heinig and Hellinger [63]. The approach described in these works produces \mathbf{T}_{n+1}^{-1} if \mathbf{T}_{n+1} is nonsingular, and the Moore-Penrose inverse \mathbf{T}_{n+1}^+ in the singular case. However, the computation required when \mathbf{T}_{n+1} is singular is several times that required in the

nonsingular case, which makes it impractical for use in the algorithms described in Chapter 6.

Numerical Properties of Toeplitz Algorithms

In this section, we will discuss the numerical properties of the fast Toeplitz algorithms, that is, the effect of finite precision calculations on the accuracy of the results. This section first briefly reviews the concepts of stability and condition as they apply to the solution of Toeplitz systems, and then examines the stability of the three algorithms discussed above.

Stability and Condition

In the context of solving linear equations, a stable algorithm is one that, when solving the equation $\mathbf{Ax} = \mathbf{b}$, produces a computed solution $\hat{\mathbf{x}}$ that is the exact solution of a nearby problem $\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}}$. By “nearby,” we mean that $\hat{\mathbf{A}}$ is close to \mathbf{A} , and $\hat{\mathbf{b}}$ is close to \mathbf{b} ; typically, a matrix norm is used to measure closeness. It is important to note that stability does *not* ensure that $\hat{\mathbf{x}}$ is close to \mathbf{x} ; this requires not only that the algorithm be stable, but also that the original problem $\mathbf{Ax} = \mathbf{b}$ be well conditioned.

A system is well conditioned when small changes in \mathbf{A} and \mathbf{b} result in small changes in the exact solution \mathbf{x} ; conversely, a system is ill conditioned when small changes in \mathbf{A} and \mathbf{b} can cause large changes in the exact solution. Because rounding errors are almost unavoidable, an ill-conditioned system is difficult to solve accurately with any algorithm, even a very stable one; once a small amount of error has crept into the computation, the system has effectively been perturbed, and if the system is ill conditioned, the computed answer may differ dramatically from the exact solution.

A quantitative measure of how much change can occur in the solution due to a change in the inputs is provided by the condition number κ . If $\mathbf{Ax} = \mathbf{b}$ is the

exact system, and the computed solution $\mathbf{x} + \Delta\mathbf{x}$ is the exact solution to a perturbed problem $(\mathbf{A} + \Delta\mathbf{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$, then the error in the solution $\Delta\mathbf{x}$ is

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa \left(\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \right) + \mathcal{O}\left(\frac{\|\Delta\mathbf{A}\|^2}{\|\mathbf{A}\|^2}, \frac{\|\Delta\mathbf{b}\|^2}{\|\mathbf{b}\|^2} \right),$$

so the output error is no more than κ times the size of the perturbations, as long as the perturbations are small enough that the higher-order terms may be ignored. The condition number κ measures the amount by which errors in the input may be magnified in the solution; if ϵ is the relative size of the error introduced by a floating-point computation², then even a perfectly stable algorithm can be expected to compute a solution with a relative error of $\kappa\epsilon$. A thorough discussion of the concepts of stability and condition has been given by Bunch [64], among many others.

Stability of Fast Toeplitz Algorithms

The stability of the conventional $\mathcal{O}(n^2)$ methods for solving Toeplitz systems of equations has been considered by Cybenko [65] and Bunch [66]. Because the conventional Levinson-Durbin and Schur algorithms solve a series of successively larger Yule-Walker problems, both of these algorithms will be unstable whenever any of the leading principal submatrices of \mathbf{T}_n is ill conditioned [65]. It is possible for a leading principal submatrix to be ill conditioned even though \mathbf{T}_n is well conditioned; for

²For most modern computers, a single elementary floating-point operation produces results that are as close as possible to the exact answer, given the limitations of floating-point representation. In this case, ϵ is the smallest number such that $1 + \epsilon \neq 1$, when computed in floating point; for 64-bit double precision conforming to the IEEE 754 standard, $\epsilon \approx 2.2 \times 10^{-16}$.

example:

$$\frac{1}{7} \begin{bmatrix} 1 & 3 & 1-\epsilon & -1 \\ 3 & 1 & 3 & 1-\epsilon \\ 1-\epsilon & 3 & 1 & 3 \\ -1 & 1-\epsilon & 3 & 1 \end{bmatrix}$$

is well conditioned, with $\kappa \approx 2.6$, but the leading 3×3 principal submatrix has condition number $\kappa \approx 5.8/\epsilon$ [67]. This indefinite matrix is solved accurately by general $\mathcal{O}(n^3)$ algorithms, but not by the fast Toeplitz algorithms.

For one class of matrices, it is possible to be certain that all of the leading principal submatrices are at least as well conditioned as \mathbf{T}_n itself. If \mathbf{T}_n is positive definite, then the interlacing property of the eigenvalues of leading principal submatrices guarantees that no leading principal submatrix will be more ill conditioned than \mathbf{T}_n .

Stability of Superfast Toeplitz Algorithms

We have seen that the fast Toeplitz algorithms are unstable whenever one of the smaller problems solved in the course of the computation is ill conditioned. The superfast algorithms have similar characteristics, but the manner in which the problem is divided is different, so the subproblems whose condition determines the stability of the algorithm are also different. Each stage of the generalized Schur algorithm solves two Schur problems of half the size of the input problem. The subdivision of the problem continues until the original problem of size n has been divided into n/N_T smaller problems, which are then solved by the standard Schur algorithm. The generalized Schur algorithm will therefore be unstable if one or more of these subproblems is ill conditioned.

One of these subproblems is the Schur problem of size N_T determined by the leading principal submatrix of dimension N_T ; the other problems are functions of

both the input vectors and intermediate results, and are not easily related to the entries of \mathbf{T}_n . A brief analysis of the stability of superfast algorithms has been given by Bunch [66], which indicates that the generalized Schur algorithm may be unstable if \mathbf{T}_{n+1} is not positive definite.

It is interesting to note that the superfast algorithms remain $\mathcal{O}(n \log^2 n)$ even if a stable $\mathcal{O}(N_T^3)$ method is used to solve the final series of subproblems. This does not render the superfast algorithms stable, because it is still possible for a subproblem of size N_T to be ill conditioned. It does, however, remove the possibility of ill conditioning for the submatrices of the $N_T \times N_T$ systems solved at the lowest level, and make it possible for the condition of the subproblem to be bounded exactly (see equation (6.5)). A more practical strategy would be to employ a fast $\mathcal{O}(N_T^2)$ method with the capability to “step over” nearly singular submatrices, such as those given by Chan and Hansen [68] and Zarowski [69]. Although extensive numerical tests have shown that loss of accuracy due to ill conditioning of subproblems is rare, this approach is available for those situations where it proves to be a problem.

Although determining beforehand whether the superfast algorithms are stable for a given input is difficult, it is simple to determine during a computation whether the problem being solved is ill conditioned. A poorly conditioned problem at any step of the Schur algorithm will be indicated by a small prediction error E_i or a large result a for that step; since the Schur algorithm is the only potentially unstable element of the generalized Schur algorithm, this provides a reliable test for ill conditioning. For the problems to be addressed in the following chapter, condition number estimation is not necessary, but in cases where a condition number estimate is needed, it may be computed from the prediction error sequence [70]. This problem is considered in greater detail in the next chapter.

CHAPTER 6

FAST ALGORITHMS FOR TOEPLITZ EIGENDECOMPOSITION

This chapter describes the application of the fast Toeplitz solution methods developed in the previous chapter to the calculation of eigenvalues and eigenvectors. The subspace techniques all require knowledge of the eigenvectors and eigenvalues of the estimated correlation matrix $\hat{\mathbf{R}}$; when a non-Toeplitz autocorrelation estimate is used, this requires $\mathcal{O}(M^3)$ operations (where M is the size of $\hat{\mathbf{R}}$), a heavy computational burden for large M . This has been a barrier to the use of subspace techniques in situations where large autocorrelation matrices are needed to achieve high accuracy.

Unfortunately, as we have seen in Chapter 4, when the data records are long, a large autocorrelation matrix is required to approach the statistical limits to frequency estimation accuracy. The effects of using Toeplitz autocorrelation estimates were also examined, and it was shown that, although the performance of estimators using a Toeplitz matrix was usually somewhat poorer than the standard (non-Toeplitz) implementations for constant matrix dimension M , acceptable results could still be obtained, particularly when the size of the input data record was large. The fast Toeplitz solution methods described in the preceding chapter make possible fast algorithms for the eigenanalysis of Toeplitz matrices. These allow the computation of any N eigenvalues and eigenvectors of a $M \times M$ Toeplitz autocorrelation estimate $\hat{\mathbf{R}}$ in $\mathcal{O}(NM^2)$ or $\mathcal{O}(NM \log^2 M)$ operations, depending on the choice of Toeplitz solution method.

When these fast Toeplitz eigendecomposition techniques are employed in subspace algorithms for frequency estimation, much larger autocorrelation matrices can be used. In many cases, the improvement in frequency estimation performance due

to the larger matrix more than compensates for the degradation caused by the use of a Toeplitz estimate of the autocorrelation matrix, so these fast implementations offer important computational advantages.

The Standard Eigenproblem

We will begin with a discussion of fast algorithms for the standard Toeplitz eigenproblem $\mathbf{T}_M \mathbf{e} = \lambda \mathbf{e}$, which is the basis for subspace algorithms for frequency estimation in white noise. The extension of this approach to the generalized Toeplitz eigenproblem $\mathbf{T}_M \mathbf{e} = \lambda \Sigma_M \mathbf{e}$ is straightforward, and will be described in the last section.

Fast Computation of the Minimum Eigenvalue

Using conventional fast algorithms for solving Toeplitz systems, Cybenko and Van Loan [71] developed an $\mathcal{O}(M^2)$ approach for finding the minimum eigenvalue of a $M \times M$ symmetric Toeplitz matrix \mathbf{T}_M . Similar approaches have also been proposed by Hayes and Clements [72], and by Hu and Kung [73]. These methods for fast Toeplitz eigendecomposition are based on the same strategy: the direct search for the zeros of the characteristic polynomial $\det(\lambda \mathbf{I} - \mathbf{T}_M)$, which are the eigenvalues of \mathbf{T}_M . The determinant is computed without forming the characteristic polynomial; explicit calculation of the characteristic polynomial leads to methods with poor numerical properties.

For general matrices, computation of the determinant requires $\mathcal{O}(M^3)$ operations, and since several evaluations of the determinant are usually needed to locate a zero, this method is almost always less efficient than other $\mathcal{O}(M^3)$ techniques, such as Householder reduction and QR iteration. However, for structured matrices whose determinants are easier to compute, a direct search for zeros of the determinant is often an effective technique, especially when only a subset of the eigenvalues are needed.

In fact, Cybenko and Van Loan's algorithm for Toeplitz eigendecomposition, which is described below, is strikingly similar to the bisection method [74], which is widely used for computing a subset of the eigenvalues of a symmetric tridiagonal matrix.

All of the fast Toeplitz eigendecomposition methods are based on the same properties of the prediction error for the Yule-Walker problem; we will begin with a description of the Cybenko-Van Loan algorithm, which is the basis for all of the algorithms considered here. By definition, the minimum eigenvalue λ_M , and its associated eigenvector \mathbf{e}_M satisfy the equation

$$\mathbf{T}_M \mathbf{e}_M = \lambda_M \mathbf{e}_M.$$

We will assume that λ_M is less than the minimum eigenvalue σ_{M-1} of \mathbf{T}_{M-1} , the largest leading principal submatrix of \mathbf{T}_M . Now partition \mathbf{T}_M and \mathbf{e}_M into

$$\begin{bmatrix} t_0 & \mathbf{t}_1^T \\ \mathbf{t}_1 & \mathbf{T}_{M-1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda_M \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

The restriction that $\lambda_M < \sigma_{M-1}$ implies that $\mathbf{x} \neq 0$, because if $\mathbf{x} = 0$, $\mathbf{T}_{M-1}\mathbf{y} = \lambda_M\mathbf{y}$, which, together with the interlacing property of eigenvalues, requires $\lambda_M = \sigma_{M-1}$. (For autocorrelation estimates computed from data with random noise, the eigenvalues are strictly separated with probability one, so the requirement that $\lambda_M < \sigma_{M-1}$ is satisfied with probability one.) Separating the equations gives

$$\mathbf{T}_{M-1}\mathbf{y} + \mathbf{t}_1\mathbf{x} = \lambda_M\mathbf{y}, \tag{6.1}$$

$$\mathbf{t}_1^T\mathbf{y} + t_0\mathbf{x} = \lambda_M\mathbf{x}. \tag{6.2}$$

From equation (6.1):

$$\mathbf{y} = -(\mathbf{T}_{M-1} - \lambda_M \mathbf{I})^{-1} \mathbf{t}_1 x \quad (6.3)$$

if $(\mathbf{T}_{M-1} - \lambda_M \mathbf{I})^{-1}$ exists. Let \mathbf{a} be the solution to

$$(\mathbf{T}_{M-1} - \lambda_M \mathbf{I})\mathbf{a} = -\mathbf{t}_1$$

which exists whenever a solution to equation (6.3) exists. Then

$$\mathbf{a} = -(\mathbf{T}_{M-1} - \lambda_M \mathbf{I})^{-1} \mathbf{t}_1. \quad (6.4)$$

Substituting equation (6.4) and equation (6.3) into equation (6.2),

$$\mathbf{t}_1^T \mathbf{a} x + t_0 x = \lambda_M x.$$

Since $x \neq 0$,

$$\mathbf{t}_1^T \mathbf{a} + t_0 = \lambda_M.$$

We can see that the any eigenvalue of \mathbf{T}_M is a root of the equation

$$E(\lambda) = t_0 - \lambda + \mathbf{t}_1^T \mathbf{a},$$

where $E(\lambda)$ is the final prediction error for the Yule-Walker system

$$(\mathbf{T}_M - \lambda \mathbf{I}) \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E(\lambda) \\ \mathbf{0} \end{bmatrix}.$$

From the properties of the prediction error discussed in Chapter 5,

$$E(\lambda) = \frac{\det(\mathbf{T}_M - \lambda\mathbf{I})}{\det(\mathbf{T}_{M-1} - \lambda\mathbf{I})},$$

so the roots of $E(\lambda)$ are the eigenvalues of \mathbf{T}_M , and the poles of $E(\lambda)$ are the eigenvalues of \mathbf{T}_{M-1} , as long as the eigenvalues of \mathbf{T}_M and \mathbf{T}_{M-1} are strictly separated. The computation of the ratio of the determinant of \mathbf{T}_M to that of its largest leading principal submatrix, rather than the determinant of \mathbf{T}_M itself, is also advantageous from a computational standpoint when \mathbf{T}_M has a group of closely spaced eigenvalues [74]. In this case $\det(\mathbf{T}_M - \lambda\mathbf{I}) = \prod_{k=1}^M (\lambda_k - \lambda)$ can be extremely small for values of λ between the eigenvalues in the group, causing difficulty in determining the locations of the zeros accurately. On the other hand, if the eigenvalues of \mathbf{T}_{M-1} strictly separate those of \mathbf{T}_M , $E(\lambda)$ has a pole between each pair of roots, and numerical root location is greatly simplified. It might appear that the possibility of $E(\lambda)$ becoming infinite is a drawback to this approach, however, in floating-point computations it is not unusual for the values of polynomials of high degree to exceed the maximum value allowed by the floating-point system being used, so the determinant is often effectively infinite in any case.

In finding the roots of $E(\lambda)$, the derivative with respect to λ is also useful:

$$E'(\lambda) = -1 - \mathbf{a}^T \mathbf{a}.$$

The fact that $E'(\lambda)$ is always negative further simplifies the search for roots of $E(\lambda)$. As we shall see below, the use of derivative information and the proper choice of a numerical zero-locating technique can minimize the computation required to find the eigenvalues of \mathbf{T}_M .

Cybenko and Van Loan proposed the use of bisection and Newton's method to determine the zero of $E(\lambda)$ between 0 and σ_{M-1} , which is λ_M , the smallest eigenvalue of \mathbf{T}_M . Their algorithm can be summarized as follows:

ALGORITHM 6.1: CYBENKO-VAN LOAN ALGORITHM

```

find  $\lambda^{(0)}$  such that  $\lambda_n < \lambda^{(0)} < \sigma_{n-1}$ 
while  $|E(\lambda^{(k-1)})| > \epsilon$ 
    solve  $(\mathbf{T}_{M-1} - \lambda^{(k-1)}\mathbf{I})\mathbf{a} = -\mathbf{t}$ 
     $\lambda^{(k)} = \lambda^{(k-1)} - E(\lambda^{(k-1)})/E'(\lambda^{(k-1)})$ 
end

```

Although Cybenko and Van Loan employed the Levinson-Durbin algorithm, it was noted in Chapter 5 that solution of $(\mathbf{T}_{M-1} - \lambda^{(k)}\mathbf{I})\mathbf{a} = -\mathbf{t}$ can be accomplished either by a conventional fast algorithm, or by a superfast technique, since both compute the same solution and the same sequence of prediction errors. In addition, the associated eigenvector is easily obtained once a zero λ_i of $E(\lambda)$ has been found. The eigenvector is

$$\mathbf{e} = \begin{bmatrix} x \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} x \\ \mathbf{a}x \end{bmatrix},$$

so the normalized eigenvector associated with λ_i is

$$\mathbf{e}_i = \frac{1}{\sqrt{1 + \mathbf{a}^T \mathbf{a}}} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix}.$$

If the number of iterations necessary for λ to converge to λ_i is independent of M (as is found to be the case in practice), then Algorithm 6.1 is an $\mathcal{O}(M^2)$ or $\mathcal{O}(M \log^2 M)$ method, depending on the choice of Toeplitz solution algorithm, for finding an eigenvalue and the associated eigenvector of a symmetric Toeplitz matrix.

Fast Computation of Any Eigenvalue and Eigenvector

The approach used by Cybenko and Van Loan was employed by Trench [75,76], and later by Noor and Morgera [77], in an $\mathcal{O}(M^2)$ method for finding any eigenvalue and its associated eigenvector. The Cybenko-Van Loan algorithm may be used without modification for finding any eigenvalue; however, doing so requires solving indefinite Toeplitz systems, and the accuracy of the answers may therefore be affected by the instability of the solution algorithms. Several approaches to resolving this difficulty are discussed in a later section.

The procedure described in this chapter is a further development of the approach taken by Trench, with several improvements to enhance the efficiency and reliability of the algorithm. These include the use of a superfast Toeplitz solver to reduce the asymptotic complexity of the algorithm to $\mathcal{O}(M \log^2 M)$, the development of a test to detect whether the computed results have been corrupted by numerical instability, and the extension of the algorithm to solve the symmetric definite Toeplitz generalized eigenproblem.

In the following sections, the improved Toeplitz eigendecomposition algorithm for the standard eigenproblem $\mathbf{T}_M \mathbf{e} = \lambda \mathbf{e}$ will be described. The algorithm proceeds in two stages: the desired eigenvalue is first bracketed, then a numerical root-finding procedure is used to locate the eigenvalue exactly. A detailed discussion of each of these stages is given in the sections below.

Bracketing the Desired Eigenvalue

It was shown above that the roots of $E(\lambda)$ are the eigenvalues of \mathbf{T}_M ; but before a root-finding procedure can be employed to locate a specific eigenvalue, a bracketing interval must be found that contains only the desired root λ_i . Because eigenvalues of the largest leading principal submatrix \mathbf{T}_{M-1} correspond to poles of the

function $E(\lambda)$, it is also desirable that the bracketing interval exclude all eigenvalues of \mathbf{T}_{M-1} .

Brackets on the location of a desired eigenvalue may be easily determined by the use of the prediction error sequence resulting from solving the Yule-Walker equation defined by $\mathbf{T}_M - \lambda \mathbf{I}$. This procedure is sometimes referred to as “slicing” the spectrum of \mathbf{T}_M [78], because each solution for a given λ determines the number of eigenvalues above and below λ . An illustration of the effect of one “slice” on the bracketing intervals for each eigenvalue is given by Figure 6.1 below, where the vertical lines indicate the bracketing intervals for each eigenvalue.

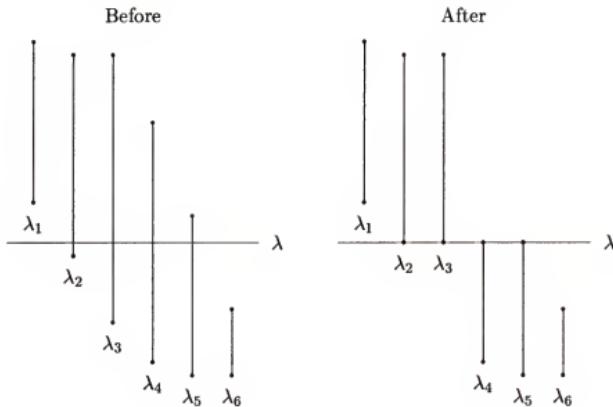


Figure 6.1: Effect of “Slicing” the Spectrum at a Single Value of λ on the Eigenvalue Bracket Intervals

A simple bisection procedure is used to compute a bracketing interval for the desired eigenvalue; the number of eigenvalues of \mathbf{T}_M that are greater than λ is found by solving the Yule-Walker problem for $\mathbf{T}_M - \lambda \mathbf{I}$, and counting the number of prediction errors greater than zero. Because the first $M - 1$ prediction errors correspond to the matrix $\mathbf{T}_{M-1} - \lambda \mathbf{I}$, the results of the same calculation may also be

used to ensure that no eigenvalue of \mathbf{T}_{M-1} is included in the interval. The procedure is summarized in Algorithm 6.2.

ALGORITHM 6.2: EIGENVALUE BRACKETING

```

Determine initial brackets  $l_i \leq \lambda_i \leq u_i$ 
 $k_1 = M$  (number of eigenvalues of  $\mathbf{T}_M$  less than  $l_i$ )
 $m_1 = M - 1$  (number eigenvalues of  $\mathbf{T}_{M-1}$  less than  $l_i$ )
 $k_2 = 0$  (number of eigenvalues of  $\mathbf{T}_M$  greater than  $u_i$ )
 $m_2 = 0$  (number of eigenvalues of  $\mathbf{T}_{M-1}$  greater than  $u_i$ )
 $\alpha = t(0)$ 
while  $k_1 - k_2 > 1$  or  $m_1 \neq m_2$ 
     $\lambda = (l_i + u_i)/2$ 
     $t(0) = \alpha - \lambda$ 
    Solve the Yule-Walker problem  $\mathbf{T}_{M-1}\mathbf{a} = -\mathbf{t}$ 
     $m = M - 1 -$  (number of negative  $e(j)$  for  $1 \leq j \leq M - 1$ )
     $k = M -$  (number of negative  $e(j)$  for  $1 \leq j \leq M$ )
    if  $k < i$ 
         $u_i = \lambda$ 
         $k_2 = k$ 
         $m_2 = m$ 
    else
         $l_i = \lambda$ 
         $k_1 = k$ 
         $m_1 = m$ 
    end
end

```

Effects of Instability on the Bracketing Process

In light of the fact that the solution algorithm used to calculate the prediction errors is unstable when λ is very close to an eigenvalue of one of the leading principal submatrices of \mathbf{T}_M , it is clear that we must examine the effects of instability on the accuracy of the bracketing process. There are $M(M - 1)/2$ eigenvalues of the leading principal submatrices of \mathbf{T}_M , distributed between λ_1 and λ_M ; each of these is surrounded by a small region where solution of $(\mathbf{T}_M - \lambda\mathbf{I})\mathbf{a} = -\mathbf{t}$ using fast or superfast algorithms is unstable.

It should be remarked that numerical experiments have shown that the probability of an evaluation point falling within one of these unstable regions is extremely small, at least for the types of matrices encountered in subspace estimation. During the computation of hundreds of thousands of eigenvalues and eigenvectors of randomly generated autocorrelation matrices, with dimensions ranging from 17 to 16385, failure of the bracketing algorithm has never occurred. Nevertheless, it is clearly desirable to have a test for instability to avoid any possibility of error in the bracketing process, and it is essential that the test be efficient.

In fact, several efficient tests do exist. The simplest of these forms the first line of defense against errors in the bracketing process; it is based on consistency checking of the results of the bracket computations. Suppose the slicing procedure (solution of the Yule-Walker problem defined by $\mathbf{T}_M - \lambda \mathbf{I}$) is carried out at three different values of λ , $\lambda_a > \lambda_b > \lambda_c$, with resulting prediction error sequences E_a , E_b , and E_c . If we define $N_{neg}(E(1:k))$ to be the number of negative entries among the first k elements of E , then the interlacing property requires that $N_{neg}(E_a(1:k)) \leq N_{neg}(E_b(1:k)) \leq N_{neg}(E_c(1:k))$ for any k between 0 and M . If E is stored when each bracket point is calculated, it is simple to test the consistency of the results of each evaluation. In practice, storage of the full prediction error vector for each bracket point is often impractical, since this requires $2M^2$ memory locations. However, to compute the desired bracket interval, the values $N_{neg}(E(1:M-1))$ and $N_{neg}(E(1:M))$ must be stored anyway, and these can form the basis for a useful test that requires almost no additional computation.

A second type of test attempts to estimate the condition of each problem as it is solved. As discussed earlier, any computational method that divides a problem into smaller pieces faces the possibility that, although the problem as a whole is well conditioned, the pieces may not be. This possibility exists for both the fast and superfast algorithms; only the way the problem is divided is different. A test that

determines the condition of each subproblem can detect when the results of a bracket computation may be in error.

In the fast algorithms, the Levinson-Durbin algorithm is used to compute the E_i , and it is unstable only if one of the leading principal submatrices of \mathbf{T}_M is ill conditioned [65]. If \mathbf{T}_M is a Toeplitz matrix, the largest leading principal submatrix \mathbf{T}_{M-1} is nonsingular, and

$$\mathbf{T}_M \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E_{M-1} \\ \mathbf{0} \end{bmatrix},$$

with $E_{M-1} \neq 0$, then Theorem 5.1 shows that

$$\mathbf{T}_M^{-1} = \frac{1}{E_{M-1}}(\mathbf{L}\mathbf{L}^T - \mathbf{M}\mathbf{M}^T).$$

Using this and the facts that $\|\mathbf{L}\|_1 = \|\mathbf{L}^T\|_1 = 1 + \|\mathbf{a}\|_1$ and $\|\mathbf{M}\|_1 = \|\mathbf{M}^T\|_1 = \|\mathbf{a}\|_1$, it is simple to bound $\|\mathbf{T}_M^{-1}\|_1$:

$$\begin{aligned} \|\mathbf{T}_M^{-1}\|_1 &= \left\| \frac{1}{E_{M-1}}(\mathbf{L}\mathbf{L}^T - \mathbf{M}\mathbf{M}^T) \right\|_1 \\ &= \frac{1}{|E_{M-1}|} \|(\mathbf{L}\mathbf{L}^T - \mathbf{M}\mathbf{M}^T)\|_1 \\ &\leq \frac{1}{|E_{M-1}|} \left[(1 + \|\mathbf{a}\|_1)^2 + \|\mathbf{a}\|_1^2 \right] \\ &= \frac{1 + 2\|\mathbf{a}\|_1 + 2\|\mathbf{a}\|_1^2}{|E_{M-1}|} \end{aligned}$$

The 1-norm condition of \mathbf{T}_M is therefore bounded by ¹

$$\kappa_1(\mathbf{T}_M) \leq \frac{\|\mathbf{T}_M\|_1(1 + 2\|\mathbf{a}\|_1 + 2\|\mathbf{a}\|_1^2)}{|E_{M-1}|}, \quad (6.5)$$

and a very similar derivation can be used to show that this bound also applies to $\kappa_\infty(\mathbf{T}_M)$. Because $\|\mathbf{T}_M\|_1$ is easily computed, this provides a simple test for ill-conditioning of the subproblems solved by the conventional fast algorithms; a notable advantage of this approach is that the additional computation required is negligible, and may be performed during the solution of each subproblem.

For the superfast generalized Schur algorithms, the problem is recursively subdivided into smaller subproblems, each of which is solved by the same algorithm. The generalized Schur algorithm for solving the Yule-Walker problem is unstable only if one of the subproblems is ill conditioned. As in the case of the Levinson-Durbin algorithm, ill-conditioning of a subproblem for the Schur algorithm is often indicated by small prediction errors and a large solution vector. The quantity $\|\mathbf{T}_M\|(1 + 2\|\mathbf{a}\|_1 + 2\|\mathbf{a}\|_1^2)/|E_{M-1}|$ may be still used as an estimator of the stability of the solution, although it is no longer a rigorous bound for the condition number.

By examining the prediction errors and solution vectors returned by the solution algorithm, it is possible to detect ill-conditioning of a subproblem. Since the bracketing process does not require specific values of λ , but only values that decrease the bracketing interval significantly, the simplest solution to instability during the bracketing phase is to discard the results of a poorly-conditioned evaluation, and choose a new λ displaced from the original value by a small fraction of the width of the bracketing interval.

¹Both Cybenko [65] and Koltracht and Lancaster [70] have developed somewhat tighter bounds on $\|\mathbf{T}_M\|$ than those given here, for the case when \mathbf{T}_M is positive definite. In particular, Koltracht and Lancaster show that the term $(1 + 2\|\mathbf{a}\|_1 + 2\|\mathbf{a}\|_1^2)$ in equation (6.5) may be replaced by $(1 + \|\mathbf{a}\|_1^2)$ if \mathbf{T}_M is positive definite. They also show that this is a tighter bound than those given by Cybenko. However, the bound given by (6.5) is only a factor of two looser, and applies to indefinite Toeplitz matrices as well.

The tests described above, although they are not designed to totally exclude the possibility of an error in bracketing, greatly reduce the possibility that an erroneous bracket interval will occur. If, despite these safeguards, an error occurs, it will be discovered when the root-location phase fails to converge to an eigenvalue, and only the amount of computation necessary to locate a single eigenvalue will have been wasted. These tests do not address the somewhat more difficult problem of determining the accuracy of the eigenvalues and eigenvectors computed during the second phase; rigorous tests for the accuracy of these computations are described in the next chapter.

Enhancing the Efficiency of the Bracketing Stage

Use of bisection to bracket the desired eigenvalue is a common feature of all earlier Toeplitz eigensolvers. In addition to stability verification techniques for the bracketing phase, two further enhancements have been developed in this work to improve its efficiency and accuracy. First, to reduce the number of Yule-Walker solutions needed to isolate a root, an initial bracketing interval is constructed by embedding the $M \times M$ Toeplitz matrix \mathbf{T}_M in a circulant matrix \mathbf{C} . The eigenvalues of a circulant matrix are given by the Fourier transform of its first column, so they can be computed in $\mathcal{O}(M \log M)$ operations; then, since \mathbf{T}_M is a leading principal submatrix of \mathbf{C} , the interlacing property may be used to compute upper and lower limits for each eigenvalue of \mathbf{T}_M . (This approach was suggested by Feyh and Mullis [79], in the context of a method for solving the Toeplitz inverse eigenvalue problem, that is, finding a Toeplitz matrix with a given set of eigenvalues.) These results are used to initialize a table of brackets for each eigenvalue. Bounds for the minimum and maximum eigenvalue of \mathbf{T}_M are also computed using the procedure given by Hertz [80]. These computations are performed each time a new matrix is input for solution.

Second, each time the Yule-Walker problem is solved for a specific value of λ , the bracket table is updated. Because the bisection process to isolate an eigenvalue λ_i solves the Yule-Walker problem for many nearby values of λ , maintaining a bracket table dramatically reduces the amount of computation required for locating a group of adjacent eigenvalues. In addition to the upper and lower bounds for the eigenvalues, the values of $E(\lambda)$ and $E'(\lambda)$ at the bounds are also stored, so that they may be provided as input to the root-finding routine; this saves two additional Yule-Walker solutions per root.

Locating the Roots of $E(\lambda)$

Once a root of $E(\lambda)$ has been bracketed, the second stage of the computation is to locate the root precisely, using a numerical root-finding procedure to determine its location. Many techniques have been employed for this purpose: Cybenko and Van Loan [71] employed Newton's method, Hu and Kung [73], Rayleigh quotient iteration, Trench [75], the Pegasus method (a variant of the secant method [81, p. 341]), and Noor and Morgera [77], a modified form of Rayleigh quotient iteration.

To maximize the efficiency of the computation, it is clearly necessary to minimize the number of Yule-Walker solutions. The number of solutions required is determined by the root-finding technique employed; one measure of the rapidity with which a root may be located is the asymptotic convergence rate. A convergence rate of ρ means that the error in the root location decreases asymptotically as K^ρ , where K is the number of evaluations of the function (that is, the number of solutions of the Yule-Walker equation). The asymptotic convergence rate for each of the techniques mentioned above is shown in Table 6.1.

Table 6.1: Asymptotic Convergence Rates of Root-Finding Algorithms

algorithm	convergence rate
bisection method	1.000
Pegasus method	1.642
Newton's method	2.000
rational interpolation (degree = 3)	2.000
Rayleigh quotient iteration	3.000

The fastest techniques above have extremely rapid convergence rates; for example, $\rho = 3$ implies that once asymptotic convergence has been reached, if a root location accurate to 2 significant digits has been found, the next solution will determine the root location with 6 digits of accuracy, and the following evaluation will produce a location accurate to 18 significant digits. However, evaluating a technique solely on the basis of its asymptotic convergence can be misleading: a technique with $\rho = 2$ that requires 2 fewer function evaluations to reach asymptotic convergence and locate the root to 2 significant digits will reach the accuracy limit of double precision arithmetic (17 digits) faster than the $\rho = 3$ technique. The ideal root-finding method for this application would be efficient in the initial location of the root, and also have a high asymptotic convergence rate. We will discuss the performance of the most effective techniques in the sections below.

Rayleigh Quotient Iteration

Unlike the other techniques mentioned above, Rayleigh quotient iteration is not a general technique for finding roots of functions; it is specifically an eigenvalue location technique. It also requires the solution of a general Toeplitz system, rather than the Yule-Walker system required by the other approaches. Although the additional computation required can be minimized using the Gohberg-Semencul relation, the computational burden per function evaluation is still higher than for other techniques. Despite these disadvantages, the cubic convergence rate of the Rayleigh quotient approach makes it an attractive candidate for eigenvalue location, and its

use has been proposed by several researchers [73, 77]. The basic iteration is given by Algorithm 6.3.

ALGORITHM 6.3: RAYLEIGH QUOTIENT ITERATION

```

Find initial estimates  $\lambda^{(0)}$  and  $\mathbf{e}^{(0)}$ 
 $s = 1$ 
while  $1/\sqrt{s} > \epsilon$ 
  Solve  $(\mathbf{T}_M - \lambda^{(k-1)}\mathbf{I})\mathbf{x} = \mathbf{e}^{(k-1)}$ 
   $s = \mathbf{x}^T \mathbf{x}$ 
   $\lambda^{(k)} = \mathbf{x}^T \mathbf{T}_M \mathbf{x} / s$ 
   $\mathbf{e}^{(k)} = \mathbf{x} / \sqrt{s}$ 
end

```

For almost any choice of initial values $\lambda^{(0)}$ and $\mathbf{e}^{(0)}$, Rayleigh quotient iteration will converge to a pair (λ, \mathbf{e}) composed of an eigenvalue of \mathbf{T}_M and the associated eigenvector, and the convergence will ultimately be cubic [82]. Unfortunately, there is no way to be certain that it will converge to the *desired* eigenvalue, unless it is known that $\lambda^{(0)}$ and $\mathbf{e}^{(0)}$ are closer to the desired pair than to any other. By combining bisection with Rayleigh quotient iteration, Noor and Morgera [77] produced a simple bracketing algorithm that locates a particular eigenvalue and eigenvector with the same asymptotic convergence rate as Rayleigh quotient iteration.

ALGORITHM 6.4: MODIFIED RAYLEIGH QUOTIENT ITERATION

```

Find initial bounds  $l_j \leq \lambda_j \leq u_j$ 
Find initial estimates  $\lambda_j^{(0)}$  and  $\mathbf{e}_j^{(0)}$ 
 $s = 1$ 
while  $\min(u_j - l_j, 1/\sqrt{s}) > \epsilon$ 
  Solve  $(\mathbf{T}_M - \lambda_j^{(k-1)}\mathbf{I})\mathbf{x} = \mathbf{e}_j^{(k-1)}$ 
   $s = \mathbf{x}^T \mathbf{x}$ 
   $q = \mathbf{x}^T \mathbf{T}_M \mathbf{x} / s$ 
  if  $l_j < q < u_j$ 
     $\lambda_j^{(k)} = q$ 
     $\mathbf{e}_j^{(k)} = \mathbf{x} / \sqrt{s}$ 
  else
    Update bounds  $l_j$  and  $u_j$  using the prediction error sequence
     $\lambda_j^{(k)} = (l_j + u_j)/2$ 
     $\mathbf{e}_j^{(k)} = \mathbf{e}_j^{(k-1)}$ 
  end
end

```

Noor and Morgera have shown empirically that this approach outperforms Newton's method and the Pegasus algorithm when used in a fast Toeplitz eigensolver [77]. The major disadvantage of the modified Rayleigh quotient iteration is its slow initial convergence: because it employs bisection, it locates the neighborhood of the root with only linear convergence, and then abruptly transitions to cubic convergence once the Rayleigh quotient q falls within the bounds.

Rational Polynomial Interpolation

A technique based on rational polynomial interpolation is an alternative approach to location of roots; it has asymptotic convergence nearly as rapid as Rayleigh quotient iteration, and better initial convergence properties. The high performance of this approach is a consequence of the fact that the model it assumes for $E(\lambda)$ is well matched to the actual form.

Almost all root location techniques with superlinear convergence work by using the results of function evaluations to form a model of the function in the neighborhood of a root, and using that model to select the next point at which to evaluate the function; for example, Newton's method models the function as a line tangent to the function at the most recent evaluation point. By matching the model as closely as possible to the actual form of the function, excellent performance can be achieved. For locating the roots of $E(\lambda)$, a rational polynomial is clearly an appropriate model, since $E(\lambda) = \det(\mathbf{T}_M - \lambda\mathbf{I})/\det(\mathbf{T}_{M-1} - \lambda\mathbf{I})$.

A rational polynomial approximation of a function is characterized by the degrees of the numerator and denominator polynomials; the possibilities include simple polynomial approximation (degree of the denominator is zero), inverse polynomial approximation (degree of the numerator is zero), and the case where the degrees of the numerator and denominator are equal. For approximating $E(\lambda)$, where the interlacing of the eigenvalues of \mathbf{T}_M and \mathbf{T}_{M-1} guarantees that there is a pole of $E(\lambda)$ between any two roots, an approximation with numerator and denominator of nearly equal degree is the best choice, since within any interval the difference between the number of poles and the number of zeros of $E(\lambda)$ is at most one.

The simplest function of this form is given by

$$f(\lambda) = K \frac{(\lambda - z)}{(\lambda - p_1)(\lambda - p_2)}.$$

This approximating function has four unknown coefficients; by using the values of $E(\lambda)$ and $E'(\lambda)$ at the upper and lower bracket points, these coefficients may be determined. The next function evaluation point is then given by the zero z of the approximating function. Efficient methods for performing this computation have been developed by Larkin [83], and further improved by Norton [84]; Norton's implementation has been used here to locate the roots of $E(\lambda)$.

Comparison

In numerical tests with an eigenvalue location tolerance $\epsilon = 10^{-12}$, using 17-digit arithmetic (IEEE 754 double precision), the rational function interpolation method of locating the eigenvalues was found to be somewhat more efficient than the modified Rayleigh quotient iteration used by Noor and Morgera, requiring on the average 20–30% fewer evaluations of $E(\lambda)$. It was found that the rational function method achieved faster than linear convergence almost immediately (within 2–3 function evaluations), while the modified Rayleigh quotient iteration typically required 4–6 evaluations before the Rayleigh quotient fell within the bounds and superlinear convergence began.

The relative performance of the two methods depends on the location of the desired eigenvalue. Rayleigh quotient iteration was less effective when locating an eigenvalue within a closely spaced group, and more effective when the desired eigenvalue was isolated, while the performance of rational interpolation was less sensitive to the location of nearby eigenvalues. The performance of the two techniques also depends on the precision ϵ with which eigenvalue locations were computed. Since Rayleigh quotient iteration has faster asymptotic convergence, its performance compared to rational function interpolation would improve for smaller tolerances, were it not for the lower limit set by the precision of the floating-point format ($\epsilon \approx 2.2 \times 10^{-16}$); conversely, rational function interpolation showed a larger performance advantage for larger ϵ . On a machine with higher precision arithmetic, for example, Cray 128-bit double precision, Rayleigh quotient iteration would be the method of choice for small ϵ .

For the eigenvector computations described in the remainder of this work, rational function interpolation with $\epsilon = 10^{-12}$ was employed. This was followed by a single step of inverse “iteration,” that is, the eigenvalue and eigenvector estimates $\lambda^{(0)}$

and $\mathbf{e}^{(0)}$ resulting from the rational function solution were used to define the equation $\mathbf{T}_M \mathbf{y} = \lambda^{(0)} \mathbf{e}^{(0)}$, and the result $\hat{\mathbf{e}} = \mathbf{y} / \|\mathbf{y}\|$ was taken as the eigenvector estimate.

The Chan-Hansen Algorithm: A Stable Method

In the following chapter, we will discuss tests for verifying that the eigenvalues and eigenvectors computed by these fast techniques are correct. To provide an efficient means of recalculating any that are found to be in error, it is necessary to have a stable fast technique for solving the Yule-Walker problem. An algorithm recently developed by Chan and Hansen [68] answers this need (in fact, the possibility of using this algorithm for the Toeplitz eigenvalue problem is mentioned by Chan and Hansen). The algorithm solves general Toeplitz systems with an asymptotic complexity of $4M^2$ operations, making it roughly twice as complex as the Levinson-Durbin algorithm, and the same complexity as Trench's algorithm for solving general Toeplitz systems. The fact that this algorithm solves general Toeplitz systems makes it possible to employ it for either the rational polynomial, Rayleigh quotient, or inverse iteration methods. It is of course possible to employ this algorithm exclusively during the eigenvalue and eigenvector computations, removing the need for testing, but it has been found that the rarity of errors makes it more efficient to use the faster algorithms and test the results.

The algorithm developed by Chan and Hansen, called the look-ahead Levinson algorithm, is a modified form of Trench's algorithm that estimates the condition number of each leading principal submatrix before attempting to solve it, and incorporates a LU decomposition routine that allows it to step over one or more nearly singular submatrices. The algorithm is stable for any Toeplitz matrix that does not have several consecutive nearly singular leading principal submatrices, and provides both an estimate of the condition number of the matrix and an accuracy estimate for the computed solution.

Although the LU decomposition routine is $\mathcal{O}(P^3)$, where P is the maximum number of consecutive ill-conditioned leading principal submatrices, numerical tests [68, 85] on random matrices indicate that very few LU steps are required, and that the typical overhead on matrices of order 1000–2000 is approximately 15%, compared to the standard Trench algorithm. The accuracy of the results for indefinite matrices is also reported to be better than that obtained from Trench’s algorithm. The Chan-Hansen algorithm is thus capable of solving accurately almost all of the problems that cannot be accurately solved by the fast and superfast algorithms, and provides a means of recomputing those eigenvalues and eigenvectors found to be in error by the tests described in the next chapter.

The Generalized Eigenproblem

The extension of the fast Toeplitz eigensolvers to the generalized Toeplitz eigenproblem $\mathbf{T}_M \mathbf{e} - \lambda \boldsymbol{\Sigma}_M \mathbf{e}$ is straightforward, since the techniques used for eigenvalue bracketing and location generalize with only slight modifications. The generalized eigenvalues are the roots of the equation $\det(\lambda \boldsymbol{\Sigma}_M - \mathbf{T}_M) = 0$, so, just as in the standard problem, we must locate the zeros of the prediction error function $E(\lambda)$.

The bracketing phase relies on the Sylvester inertia principle, which extends to the generalized eigenproblem without difficulty [82, p. 46]. As we have seen above, the efficiency of the location phase is greatly improved if the derivative $E'(\lambda)$ is available. The Yule-Walker problem to be solved is

$$\begin{bmatrix} t_0 - \lambda \sigma_0 & \mathbf{t}^T - \lambda \mathbf{s}^T \\ \mathbf{t} - \lambda \mathbf{s} & \mathbf{T}_{M-1} - \lambda \boldsymbol{\Sigma}_{M-1} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} E(\lambda) \\ \mathbf{0} \end{bmatrix}.$$

Separating the equations yields

$$t_0 - \lambda \sigma_0 + (\mathbf{t}^T - \lambda \mathbf{s}^T) \mathbf{a} = E(\lambda)$$

$$\mathbf{t} - \lambda \mathbf{s} + (\mathbf{T}_{M-1} - \lambda \boldsymbol{\Sigma}_{M-1}) \mathbf{a} = 0.$$

Taking the derivatives of these equations, we find that

$$E'(\lambda) = -\sigma_0 - \mathbf{s}^T \mathbf{a} + (\mathbf{t}^T - \lambda \mathbf{s}^T) \mathbf{a}',$$

and

$$\mathbf{a}' = (\mathbf{T}_{M-1} - \lambda \boldsymbol{\Sigma}_{M-1})^{-1}(\mathbf{s} + \boldsymbol{\Sigma}_{M-1} \mathbf{a}).$$

Using the fact that $(\mathbf{t}^T - \lambda \mathbf{s}^T)(\mathbf{T}_{M-1} - \lambda \boldsymbol{\Sigma}_{M-1})^{-1} = -\mathbf{a}^T$, we have

$$\begin{aligned} E'(\lambda) &= -\sigma_0 - \mathbf{s}^T \mathbf{a} + (\mathbf{t}^T - \lambda \mathbf{s}^T)(\mathbf{T}_{M-1} - \lambda \boldsymbol{\Sigma}_{M-1})^{-1}(\mathbf{s} + \boldsymbol{\Sigma}_{M-1} \mathbf{a}) \\ &= -\sigma_0 - \mathbf{s}^T \mathbf{a} - \mathbf{a}^T(\mathbf{s} + \boldsymbol{\Sigma}_{M-1} \mathbf{a}). \end{aligned}$$

The derivative can be written as

$$E'(\lambda) = - \begin{bmatrix} 1 & \mathbf{a}^T \end{bmatrix} \begin{bmatrix} \sigma_0 & \mathbf{s}^T \\ \mathbf{s} & \boldsymbol{\Sigma}_{M-1} \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a} \end{bmatrix},$$

and consequently can be computed using a fast algorithm in $\mathcal{O}(M \log M)$ operations. Because $\boldsymbol{\Sigma}_M$ is positive definite, $E'(\lambda)$ is always negative, just as in the standard case.

The initial eigenvalue bounds for the standard eigenproblem were found by embedding the Toeplitz matrix in a circulant matrix, finding the eigenvalues of the circulant matrix using the FFT, and using the interlacing property to bound the eigenvalues of the original Toeplitz matrix. The interlacing property also holds for definite matrix pencils [86, p. 340], and so exactly the same technique may be applied to find initial bounds for the generalized eigenvalues.

As before, either Rayleigh quotient iteration or inverse iteration can be used to improve the accuracy of a generalized eigenvalue and eigenvector pair. Algorithm 6.5 shows how the iteration process extends to the generalized eigenvalue problem.

ALGORITHM 6.5: GENERALIZED RAYLEIGH QUOTIENT ITERATION

```

Find initial estimates  $\lambda^{(0)}$  and  $\mathbf{e}^{(0)}$ 
 $s = 1$ 
while  $1/s > \epsilon$ 
  Solve  $(\mathbf{T}_M - \lambda^{(k-1)}\Sigma_M)\mathbf{x} = \Sigma_M\mathbf{e}^{(k-1)}$ 
   $s = \mathbf{x}^T\Sigma_M\mathbf{x}$ 
   $\lambda^{(k)} = \mathbf{x}^T\mathbf{T}_M\mathbf{x}/s$ 
   $\mathbf{e}^{(k)} = \mathbf{x}/\sqrt{s}$ 
end

```

One additional caution concerning the generalized eigenvalue problem is appropriate: since the generalized eigenvectors are not orthogonal in the usual sense, computational shortcuts based on equation (3.18) are no longer applicable. This can lead to increased computation for noise subspaces methods when $M \gg N$.

In this chapter, we have seen that the existence of fast algorithms for the solution of Toeplitz systems makes it possible to solve both the standard eigenproblem and the definite generalized eigenproblem for symmetric Toeplitz matrices or matrix pencils efficiently. A procedure has been presented for ensuring that the initial bounds computed for the eigenvalues are reliable despite the possible instability of the Toeplitz solution algorithms. In the next chapter, tests will be developed for verifying that instability has not caused errors in the eigenvalues and eigenvectors themselves.

CHAPTER 7

VERIFYING THE ACCURACY OF RESULTS

The preceding chapter describes a method for efficiently computing eigenvalues and eigenvectors of Toeplitz matrices that is based on fast algorithms for solution of the Yule-Walker problem. Although the algorithms used cannot be guaranteed stable, errors in the computed eigenvalues and eigenvectors due to instability in the solution of the Yule-Walker problem are rare. When these algorithms are used to solve a system with an ill-conditioned subproblem, for which they are unstable, the ill-conditioning of the subproblem in question will generally be revealed by a small value of the prediction error or by a solution vector that has a large norm.

Because the systems solved during the bracketing phase are rarely close to singular, and because a different evaluation point may be chosen if a particular λ is found to cause instability, it is simple to avoid regions of instability during the bracketing phase. In practice, it has been found that errors in the second phase, computation of the desired eigenvalue and eigenvector, are also very rare. However, because the process of computing the eigenvalues deliberately causes $E(\lambda)$ to become as small as possible, and $\|a\|$ to be as large as possible, the techniques described in the previous chapter for estimating subproblem condition during the bracketing phase are not useful during the eigenvector computation phase. The consistency tests can still be used, of course, and when an eigenvalue is being located exactly, it is feasible to store the entire prediction error vectors for the upper and lower bounds. These tests do not provide an indication of the accuracy of the computed results, however; tests for this purpose are described below.

An Extremely Simple Test

Before continuing to the more complicated tests that form the central focus of this chapter, it is worthwhile to mention a very simple test that is capable of detecting gross errors in the computed eigenvectors. It is well known that the eigenvectors of a symmetric $M \times M$ Toeplitz matrix are either symmetric ($e(i) = e(M - i + 1)$), or antisymmetric ($e(i) = -e(M - i + 1)$) [87]. If an estimated eigenvector fails this test, then it is obviously erroneous, and the more sophisticated tests described below are unnecessary. However, the fact that an eigenvector is symmetric or antisymmetric to within a given tolerance is not sufficient to establish that it is accurate, nor can meaningful bounds on the accuracy of the subspaces defined by a set of eigenvectors be determined solely on the basis of symmetry tests.

Many other properties of the symmetric and antisymmetric eigenvectors of Toeplitz matrices are known, but most are of little assistance in verifying results. It is known, for example, that any $M \times M$ Toeplitz matrix has $\lceil M/2 \rceil$ symmetric and $\lfloor M/2 \rfloor$ antisymmetric eigenvectors [87]. For some special cases, such as tridiagonal Toeplitz matrices, and Toeplitz matrices which are the autocorrelation matrices of stationary random processes with decreasing power spectra (that is, $R(i, j) = R(|i - j|)$ is the Fourier transform of a function $S(\omega)$ such that $S(\omega_1) > S(\omega_2)$ if $\omega_1 < \omega_2$), it has been shown that the symmetric and antisymmetric eigenvectors appear in alternating order [88]. Although this particular property does not apply in general, the question of whether some form of ordering relation applies for the symmetric and antisymmetric eigenvectors of all Toeplitz matrices is still open.

Error Detection

In this chapter, simple, efficient, and reliable tests for the accuracy of the computed eigenvalues and eigenvectors will be described. Tests are developed both for

a single eigenvalue-eigenvector pair, and for a set of eigenvalues and their associated eigenvectors, which define an invariant subspace. In the current application, we are concerned with the signal and noise subspaces, that is, subspaces that are defined by contiguous sets of eigenvalues, so the tests for subspaces will be specialized to that case.

In order to take advantage of the fast algorithms for Toeplitz matrix-vector multiplication, the accuracy tests will be carried out using a residual matrix. To illustrate the use of a residual, consider the case when the computed invariant subspaces and eigenvalues are exact: if \mathbf{R} is a $M \times M$ real symmetric matrix, the N orthonormal columns of \mathbf{E} define an invariant subspace of \mathbf{R} , and \mathbf{L} is a $N \times N$ diagonal matrix whose entries are the eigenvalues associated with the columns of \mathbf{E} , then

$$\mathbf{R}\mathbf{E} - \mathbf{E}\mathbf{L} = \mathbf{0}.$$

Now, suppose that \mathbf{E} and \mathbf{L} are not known exactly, but that we have computed an approximate $\hat{\mathbf{E}}$ and $\hat{\mathbf{L}}$. If we define a residual matrix \mathbf{D} as¹

$$\mathbf{R}\hat{\mathbf{E}} - \hat{\mathbf{E}}\hat{\mathbf{L}} = \mathbf{D},$$

then the size of \mathbf{D} , measured using an appropriate matrix norm, can be used to determine the accuracy of $\hat{\mathbf{E}}$ and $\hat{\mathbf{L}}$. Notice that if \mathbf{R} is Toeplitz, the product $\mathbf{R}\hat{\mathbf{E}}$ can be computed using the fast Fourier transform. Since N is usually much smaller than M , we are usually less concerned with the efficiency of computing the other product $\hat{\mathbf{E}}\hat{\mathbf{L}}$, but if N is large enough that this is important, it is possible to choose $\hat{\mathbf{L}}$ so that this computation is very efficient as well; these issues will be discussed in more detail below.

¹In the usual notation, the residual is denoted by \mathbf{R} ; to avoid confusion with the autocorrelation matrix, we will use \mathbf{D} to denote the residual here.

To interpret the results of this computation, it is necessary to relate the size of the residual to the accuracy of the approximate eigenvalues and the subspace. This connection is provided by a more complete form of the Sin Θ theorem first described in Chapter 3 [40].

THEOREM 7.1 (SIN Θ THEOREM) *Let \mathbf{R} be an $M \times M$ real symmetric matrix, and express the eigendecomposition of \mathbf{R} as*

$$\mathbf{R} = [\mathbf{E}_S, \mathbf{E}_N]^T \text{diag}(\lambda_1, \dots, \lambda_N, \lambda_{N+1}, \dots, \lambda_M) [\mathbf{E}_S, \mathbf{E}_N],$$

where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$, the columns of the $M \times N$ matrix \mathbf{E}_S span the exact signal subspace \mathcal{S} , and the columns of the $M \times M - N$ matrix \mathbf{E}_N span the exact noise subspace \mathcal{N} .

In addition, let $\hat{\mathbf{R}} = \mathbf{R} + \mathbf{H}$ also be a symmetric matrix, and express its eigendecomposition as

$$\hat{\mathbf{R}} = [\hat{\mathbf{E}}_S, \hat{\mathbf{E}}_N]^T \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_N, \hat{\lambda}_{N+1}, \dots, \hat{\lambda}_M) [\hat{\mathbf{E}}_S, \hat{\mathbf{E}}_N],$$

where $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_M$, the N columns of $\hat{\mathbf{E}}_S$ span the approximate signal subspace $\hat{\mathcal{S}}$, and the $M - N$ columns of $\hat{\mathbf{E}}_N$ span the approximate noise subspace $\hat{\mathcal{N}}$.

Finally, define the matrix residuals \mathbf{D}_S and \mathbf{D}_N as

$$\begin{aligned} \mathbf{D}_S &= \mathbf{R}\hat{\mathbf{E}}_S - \hat{\mathbf{E}}_S\hat{\mathbf{L}}_S, \\ \mathbf{D}_N &= \mathbf{R}\hat{\mathbf{E}}_N - \hat{\mathbf{E}}_N\hat{\mathbf{L}}_N, \end{aligned}$$

where $\hat{\mathbf{L}}_S$ and $\hat{\mathbf{L}}_N$ are symmetric matrices, the eigenvalues of $\hat{\mathbf{L}}_S$ are $\hat{\lambda}_1, \dots, \hat{\lambda}_N$, and the eigenvalues of $\hat{\mathbf{L}}_N$ are $\hat{\lambda}_{N+1}, \dots, \hat{\lambda}_M$.

Then:

$$\|\sin \Theta(\mathcal{S}, \hat{\mathcal{S}})\| \leq \frac{\|\mathbf{D}_S\|}{\hat{\lambda}_N - \hat{\lambda}_{N+1}},$$

and

$$\|\sin \Theta(\mathcal{N}, \hat{\mathcal{N}})\| \leq \frac{\|\mathbf{D}_N\|}{\lambda_N - \hat{\lambda}_{N+1}},$$

for any unitarily invariant norm.

The Sin Θ theorem allows the accuracy of a calculated subspace to be verified by examining the residual matrix \mathbf{D} . The residual can be calculated for an invariant subspace of any dimension, and provides an efficient means of testing the accuracy of complete signal or noise subspaces; later in this chapter, we will also discuss a form of this theorem that may be used to test individual eigenvalues and eigenvectors.

In order to bound the error in the subspace estimate, it is necessary to compute \mathbf{D} , compute exactly or find an upper bound for a unitarily invariant norm $\|\mathbf{D}\|$, and compute or find a lower bound for the gap between the eigenvalues associated with each of the subspaces. For this computation to be practical in the context of a fast eigendecomposition routine, it is of course essential that these computations be performed as efficiently as possible. We will consider each of these steps in turn in the following sections.

Computation of the Residual

Both of the residuals are the result of a computation of the form $\mathbf{R}\hat{\mathbf{E}} - \hat{\mathbf{E}}\hat{\mathbf{L}}$; since the computations are similar for both residuals, we will use a general notation in discussing the problem. The matrix \mathbf{R} is $M \times M$, and the matrix $\hat{\mathbf{E}}$ is either $M \times N$, for the signal subspace, or $M \times M - N$, for the noise subspace; here, the number of columns in $\hat{\mathbf{E}}$ will be denoted by N_E . To compute \mathbf{D} without taking advantage

of any special structure, $2M^2N_E$ operations are required to compute $\mathbf{R}\hat{\mathbf{E}}$, $2MN_E^2$ for $\hat{\mathbf{E}}\hat{\mathbf{L}}$, and MN_E to subtract the two resulting matrices.

Since the computation of $\mathbf{R}\hat{\mathbf{E}}$ by simple matrix multiplication requires $\mathcal{O}(M^2)$ operations, it is clear that we must perform this step more efficiently, particularly if it is to be used with the superfast methods; otherwise, checking the results of the computation will have a higher asymptotic complexity than the computation itself. We have already seen in Chapter 5 that multiplication of a vector by a Toeplitz matrix can be performed very efficiently using Algorithm 5.5. Computation of the product $\mathbf{R}\hat{\mathbf{E}}$ will require one forward FFT of length at least $2M-1$ to transform \mathbf{R} , N_E forward FFTs of the same length for the columns of $\hat{\mathbf{E}}$, MN_E complex multiplications, and N_E inverse FFTs, for a total of $2N_E + 1$ FFTs of length at least $2M-1$.

Using a split-radix FFT algorithm, the FFT of a real sequence of length $K = 2^k$ may be computed in $2K\log_2 K - 6K + 8$ operations, and the inverse FFT in $2K\log_2 K - 5K + 9$ operations [58]. The transformed vector is stored in a compact form that takes advantage of the symmetry of the Fourier transform of a real sequence, and this also reduces the number of operations required to perform the point-by-point multiplication of the transformed vectors to $MN_E/2$. Unfortunately, the least efficient case occurs when $M = 2^j + 1$, exactly the size of the \mathbf{T}_M required for the superfast eigendecomposition techniques. In this case, $2M-1 = 2^{j+1}+1$, and $K = 2^{j+2} \approx 4M$. Despite this, the computation of \mathbf{D} still requires less than $(N_E + 1)(8M\log_2 M - 8M + 8) + MN_E/2 + N_E(8M\log_2 M - 4M + 9)$ operations, or about $(16N_E + 8)M\log_2 M - 11.5MN_E$. Since these algorithms are intended for use when $M \gg N_E$, the reduction from $\mathcal{O}(M^2N_E)$ to $\mathcal{O}(MN_E \log M)$ is very significant.

The next step to be considered is the computation of $\hat{\mathbf{E}}\hat{\mathbf{L}}$. In Theorem 7.1, the only restriction placed on $\hat{\mathbf{L}}$ was that its eigenvalues be known. Since eigenvalue estimates are available from the eigenvector computation, one possibility is to choose

a $\hat{\mathbf{L}}$ that has these estimates as its eigenvalues. The simplest choice is

$$\begin{aligned}\hat{\mathbf{L}}_S &= \text{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_N), \\ \hat{\mathbf{L}}_N &= \text{diag}(\hat{\lambda}_{N+1}, \dots, \hat{\lambda}_M),\end{aligned}$$

where the $\hat{\lambda}$ are those computed along with the eigenvectors. In this case only MN_E operations are needed to compute the product $\hat{\mathbf{E}}\hat{\mathbf{L}}$.

This approach has the advantage of efficiency, but by choosing $\hat{\mathbf{L}}$ to have a special form, we can in certain cases produce a tighter bound on the error in the subspaces, with little additional computation. This is the consequence of a companion to the Sin Θ theorem, the Tan Θ theorem [40].

THEOREM 7.2 (TAN Θ THEOREM) *If \mathbf{R} and $\hat{\mathbf{R}}$, their eigendecompositions, their invariant subspaces, and the residuals \mathbf{D}_S and \mathbf{D}_N are as defined in Theorem 7.1, and if, in addition,*

$$\hat{\mathbf{L}}_S = \hat{\mathbf{E}}_S^T \mathbf{R} \hat{\mathbf{E}}_S,$$

and

$$\hat{\mathbf{L}}_N = \hat{\mathbf{E}}_N^T \mathbf{R} \hat{\mathbf{E}}_N,$$

then

$$\|\tan \Theta(\mathcal{S}, \hat{\mathcal{S}})\| \leq \frac{\|\mathbf{D}_S\|}{\hat{\lambda}_N - \hat{\lambda}_{N+1}},$$

and

$$\|\tan \Theta(\mathcal{N}, \hat{\mathcal{N}})\| \leq \frac{\|\mathbf{D}_N\|}{\lambda_N - \hat{\lambda}_{N+1}},$$

for any unitarily invariant norm.

The penalty paid for this tighter bound is the cost of computing the generalized Rayleigh quotient $\hat{\mathbf{E}}^T \mathbf{R} \hat{\mathbf{E}}$, and determining its eigenvalues. Computation of $\hat{\mathbf{E}}^T \mathbf{R} \hat{\mathbf{E}}$ merely requires multiplying $\hat{\mathbf{E}}^T$ by the previously computed $\mathbf{R} \hat{\mathbf{E}}$, at a cost of $2MN_E^2$ operations. It is then necessary to determine the largest or smallest eigenvalue of $\hat{\mathbf{L}}$ (depending on whether $\hat{\mathbf{E}}$ is a signal or noise subspace), since these are the $\hat{\lambda}$ that must be used in the denominator of the bounds. The matrix $\hat{\mathbf{L}}$ is $N_E \times N_E$, and if N_E is small, it may be practical to compute the eigenvalues by standard methods. Although this requires $4N_E^3/3$ operations, if $N_E \ll M$ the cost may still be acceptable. For example, if $N_E = 20$, the cost is approximately 10,000 operations, which is negligible compared to the cost of computing a single eigenvector when $M > 500$.

In cases where this computation is impractical, it is still possible to employ the Tan Θ theorem if $\hat{\mathbf{L}}$ is nearly diagonal. By using the Gershgorin circle theorem, it is simple to produce bounds for the largest and smallest eigenvalues of $\hat{\mathbf{L}}$. If $\hat{\mathbf{E}}$ is a good approximation to the true invariant subspace \mathbf{E} , then the generalized Rayleigh quotient $\hat{\mathbf{L}}$ will be very close to diagonal, so that the Gershgorin bounds will be very close to the actual extremal eigenvalues of $\hat{\mathbf{L}}$.

Computing the Norm of the Residual

The Sin Θ and related theorems require that $\|\sin \Theta\|$ and $\|\mathbf{D}\|$ be computed using a unitarily invariant norm. The choice of norm affects both our interpretation of the bound on $\|\sin \Theta\|$, and the amount of computation required to compute $\|\mathbf{D}\|$. The interpretation of the effects of inaccuracy of subspace estimates is simplest when the 2-norm is used, so this is the norm we will employ. This means that the calculations will yield a bound for $\|\sin \Theta\|_2$, and that we must either explicitly calculate, or find an upper bound for, $\|\mathbf{D}\|_2$.

Any unitarily invariant norm may be computed from the singular values of \mathbf{D} , but since computing the singular values requires $\mathcal{O}(MN_E^2 + N_E^3)$ operations, we will

also need a more efficient method for obtaining a unitarily invariant norm of \mathbf{D} when N_E is large. We will first consider the case where N_E is small, so that it is practical to compute the singular values explicitly.

The two standard techniques for computing the singular values of a matrix are due to Golub and Reinsch [37] and Chan [89]. The numbers of operations required for the two techniques are approximately $4MN_E^2 - 4N_E^3/3$ for the Golub-Reinsch approach, and $2MN_E^2 + 2N_E^3$ for the Chan algorithm, so that Chan's approach, also referred to as the R-SVD, is preferred when $M > 5N_E/3$. Since this is by far the most common situation in subspace estimation, the R-SVD is the preferred algorithm. Using the results of this computation, we can compute any unitarily invariant norm of \mathbf{D} , including the 2-norm, which is equal to the largest singular value.

Explicit computation of all the singular values of \mathbf{D} gives $\|\mathbf{D}\|_2$ exactly, producing the tightest bounds on the error in the subspace, but it is clear from the presence of the N_E^3 term in the operations count that this approach will become impractical if N_E is large. In these cases, one alternative is to compute the Frobenius norm $\|\mathbf{D}\|_F$, which is the square root of the sum of the squares of the elements of \mathbf{D} . Because the Frobenius norm is unitarily invariant, it is also equal to the square root of the sum of the squares of the singular values of \mathbf{D} , and therefore

$$\|\mathbf{D}\|_2 \leq \|\mathbf{D}\|_F \leq \sqrt{N_E} \|\mathbf{D}\|_2,$$

which sets an upper limit to the degradation of the subspace bound due to the use of the Frobenius norm. Computation of $\|\mathbf{D}\|_F$ requires $2MN_E$ operations, and so is much more efficient than computing the singular values. Similar bounds that require even less computation are provided by the relations

$$\|\mathbf{D}\|_2 \leq \sqrt{M} \|\mathbf{D}\|_\infty = \sqrt{M} \max_{i=1,M} \sum_{j=1}^{N_E} |D(i,j)|,$$

and

$$\|\mathbf{D}\|_2 \leq \sqrt{N_E} \|\mathbf{D}\|_1 = \sqrt{N_E} \max_{j=1, N_E} \sum_{i=1}^M |D(i, j)|,$$

for norms that require only MN_E floating-point operations to compute.

Bounding the Gap Between Subspaces

The final element in the computation of the bounds is the determination of the gap between the eigenvalues associated with the computed subspace and those associated with the exact complementary subspace; that is, $\hat{\lambda}_N - \lambda_{N+1}$ or $\lambda_N - \hat{\lambda}_{N+1}$. In Chapter 4, the locations of the eigenvalues of the exact matrix were not known, and it was necessary to use the weaker $\text{Sin } 2\Theta$ theorem. When testing the accuracy of computed results, we can use the brackets on the eigenvalues of \mathbf{R} to bound the gap:

$$\begin{aligned} \frac{\|\mathbf{D}_S\|}{\hat{\lambda}_N - \lambda_{N+1}} &\leq \frac{\|\mathbf{D}_S\|}{\hat{\lambda}_N - u_{N+1}}, \\ \frac{\|\mathbf{D}_N\|}{\lambda_N - \hat{\lambda}_{N+1}} &\leq \frac{\|\mathbf{D}_N\|}{l_N - \hat{\lambda}_{N+1}}, \end{aligned}$$

where l_N and u_{N+1} are the lower bound for λ_N and the upper bound for λ_{N+1} computed during the bracketing phase.

These bounds rely on the accuracy of the bracketing interval; they might be in error if a single value u_{N+1} or l_N is incorrect. It is possible to eliminate even this small possibility of error by computing one extra eigenvalue: either the largest of the noise eigenvalues λ_{N+1} or the smallest of the signal eigenvalues λ_N , depending on the technique being used. By finding an error bound for the computed value $\hat{\lambda}_{N+1}$ or $\hat{\lambda}_N$, we can determine the gap much more accurately, making the bounds on the subspace error tighter and assuring that no single error could cause the bound computation to fail.

The same technique may be used to assess the accuracy of any of the computed eigenvalues, although we are normally not as concerned with the accuracy of eigenvalues because most subspace techniques use them only to identify the desired invariant subspace. A residual bound for individual eigenvalues and eigenvectors is provided by the following theorem [41, p. 258].

THEOREM 7.3 *Let \mathbf{R} be a real symmetric matrix, and $(\hat{\lambda}, \hat{\mathbf{e}})$ an approximation to an eigenvalue of \mathbf{R} and the associated eigenvector. If the residual vector \mathbf{d} is defined as*

$$\mathbf{d} = \mathbf{R}\hat{\mathbf{e}} - \hat{\lambda}\hat{\mathbf{e}},$$

then there is an eigenvalue λ_i of \mathbf{R} , with associated eigenvector \mathbf{e}_i , for which

$$|\lambda_i - \hat{\lambda}| \leq \min \left\{ \|\mathbf{d}\|_2, \frac{\|\mathbf{d}\|_2^2}{\delta} \right\},$$

and

$$\sin \angle(\mathbf{e}_i, \hat{\mathbf{e}}) \leq \frac{\|\mathbf{d}\|_2}{\delta},$$

where

$$\delta = \min_{j \neq i} |\lambda_j - \hat{\lambda}|.$$

If the smaller of the intervals $(\hat{\lambda} - \|\mathbf{d}\|_2, \hat{\lambda} + \|\mathbf{d}\|_2)$, $(\hat{\lambda} - \|\mathbf{d}\|_2^2/\delta, \hat{\lambda} + \|\mathbf{d}\|_2^2/\delta)$ does not overlap two bracket intervals, then we can be certain which eigenvalue λ_i is the closest to $\hat{\lambda}$. In practice, this bound is almost always much smaller than $u_i - l_i$, so it generally produces a tighter bound on the gap, and therefore on the subspace error.

Comparing a Toeplitz Estimate with the Covariance Estimate

The sections above have been concerned with verifying that the eigenvectors and eigenvalues of a Toeplitz matrix have been computed accurately. We now turn to another question that may be answered using a residual bound: how close are the invariant subspaces of a Toeplitz estimate of the autocorrelation matrix to those of the covariance estimate? In Chapter 4, we discussed the use of a Toeplitz estimate for the autocorrelation matrix in place of the covariance estimate normally used in the subspace methods. There, it was shown that a Toeplitz estimate could produce reasonably accurate results, and, in the following chapters, it has been shown that the computations may be performed much more rapidly if a Toeplitz estimate is used.

By using a residual bound, it is possible to determine how close the computed eigenvalues and invariant subspaces of a Toeplitz estimate $\hat{\mathbf{R}}_T$ are to those of the covariance estimate $\hat{\mathbf{R}}_C$, by computing

$$\mathbf{D} = \hat{\mathbf{R}}_C \hat{\mathbf{E}} - \hat{\mathbf{E}} \hat{\mathbf{L}},$$

using the $\hat{\mathbf{E}}$ and $\hat{\mathbf{L}}$ computed from the Toeplitz matrix $\hat{\mathbf{R}}_T$. Because $\hat{\mathbf{R}}_C = \mathbf{Y}^T \mathbf{Y}$ (equation (4.6)), where \mathbf{Y} is the Toeplitz data matrix defined by equation (4.5), the matrix products may again be computed using the fast Fourier transform. However, \mathbf{Y} is $L - M + 1 \times M$, so the FFTs required are longer; the required length is at least $2L - 2M + 1$. Because two matrix-vector multiplies are necessary, the complete computation requires $4N_E + 1$ forward or inverse FFTs of this length. Because the eigenvalues of $\hat{\mathbf{R}}_C$ are not known, it is also necessary to use the residual form of the Sin 2 Θ theorem:

THEOREM 7.4 (SIN 2 Θ THEOREM) *If \mathbf{R} and $\hat{\mathbf{R}}$, their eigendecompositions, their invariant subspaces, and the residuals \mathbf{D}_S and \mathbf{D}_N are as defined in Theorem 7.1,*

then

$$\|\sin 2\Theta(\mathcal{S}, \hat{\mathcal{S}})\| \leq \frac{2\|\mathbf{D}_S\|}{\hat{\lambda}_N - \hat{\lambda}_{N+1}},$$

and

$$\|\sin 2\Theta(\mathcal{N}, \hat{\mathcal{N}})\| \leq \frac{2\|\mathbf{D}_N\|}{\hat{\lambda}_N - \hat{\lambda}_{N+1}},$$

for any unitarily invariant norm.

The resulting bound on the angle between the two subspaces makes it possible to assess not only the accuracy of the Toeplitz eigendecomposition, but also the effect of using a Toeplitz autocorrelation estimate. Numerical experiments indicate that the bound on the largest angle between the subspaces obtained by this technique is relatively tight, almost always within a factor of five of the exact maximum angle Θ_1 . Because of this, testing the accuracy in this way should lead to an accurate assessment of the effects of using a Toeplitz autocorrelation estimate.

Frequency Estimation Accuracy

The tests above produce bounds on the error in the estimated subspace $\hat{\mathbf{E}}$. It is natural to ask how the error in a subspace is related to the error in a frequency estimate derived from that subspace. For estimators such as MUSIC that compute frequency estimates as the minima of a projection onto a subspace, there is a particularly useful relation between subspace accuracy and the minimization used to find the frequency estimate. This relation is based on the following theorem [41].

THEOREM 7.5 *Let the matrices \mathbf{E} and $\hat{\mathbf{E}}$ both have N_N orthonormal columns. The orthogonal projectors onto the subspaces \mathcal{E} and $\hat{\mathcal{E}}$ spanned by their columns are $\mathbf{P} =$*

$\mathbf{E}\mathbf{E}^T$ and $\hat{\mathbf{P}} = \hat{\mathbf{E}}\hat{\mathbf{E}}^T$, respectively. Then

$$\|\mathbf{P} - \hat{\mathbf{P}}\|_2 = \|\sin \Theta(\mathcal{E}, \hat{\mathcal{E}})\|_2 = \sin \Theta_1(\mathcal{E}, \hat{\mathcal{E}}).$$

Recall from Chapter 3 that the MUSIC algorithm computes frequency estimates by locating the minima on the unit circle of the polynomial

$$\hat{p}(z) = \mathbf{z}^H \hat{\mathbf{E}}_N \hat{\mathbf{E}}_N^T \mathbf{z},$$

where $\hat{\mathbf{E}}_N$ is the matrix whose N_N columns are the eigenvectors of the noise subspace, and $\mathbf{z} = [1, z, z^2, \dots, z^{M-1}]^T$. It is simple to show that the difference between the result obtained for the exact noise subspace spanned by \mathbf{E}_N and the approximate noise subspace spanned by $\hat{\mathbf{E}}_N$ is given by

$$\begin{aligned} |\mathbf{z}^H \mathbf{E}_N \mathbf{E}_N^T \mathbf{z} - \mathbf{z}^H \hat{\mathbf{E}}_N \hat{\mathbf{E}}_N^T \mathbf{z}| &= \|\mathbf{z}^H \mathbf{E}_N \mathbf{E}_N^T \mathbf{z} - \mathbf{z}^H \hat{\mathbf{E}}_N \hat{\mathbf{E}}_N^T \mathbf{z}\|_2 \\ &\leq \|\mathbf{z}\|_2^2 \|\mathbf{E}_N \mathbf{E}_N^T - \hat{\mathbf{E}}_N \hat{\mathbf{E}}_N^T\|_2 \\ &= N_N \|\sin \Theta(\mathcal{N}, \hat{\mathcal{N}})\|_2 \\ &= N_N \sin \Theta_1(\mathcal{N}, \hat{\mathcal{N}}). \end{aligned}$$

This limits the difference between the results obtained with an “exact” set of eigenvectors and an approximate set, and can be used to bound the accuracy of the computed frequency estimate. Suppose that the result of a computation using the approximate matrix $\hat{\mathbf{E}}_N$ is a minimum $\hat{p}(\hat{\omega}) = \hat{p}_{\min}$ at the frequency $\hat{\omega}$. If it is known that $\sin \Theta_1 \leq C$, then the minimum computed using the exact matrix \mathbf{E}_N must lie between the points on either side of $\hat{\omega}$ at which $\hat{p}(\omega) = \hat{p}_{\min} + N_N C$. If the bound is obtained for the difference between the exact and computed subspaces of a Toeplitz matrix, it gives the maximum difference in the frequency estimate due to numerical error in the computation of $\hat{\mathbf{E}}_N$. If the bound is obtained for the difference between

the computed Toeplitz subspace and the subspace of the covariance estimate, it gives the maximum difference due not only to numerical error, but also to the use of a Toeplitz estimate of \mathbf{R} .

Verifying Generalized Eigendecompositions

Extensions of the Sin Θ and Sin2 Θ theorems to the generalized eigenvalue problem have been developed by Sun [90, 91]. Unfortunately, these extensions include only perturbation forms of these theorems, not the residual forms required for efficiently verifying the accuracy of generalized eigendecompositions. One bound that still holds is a generalized form of Theorem 7.3: for any approximate generalized eigenvalue and eigenvalue $(\hat{\lambda}, \hat{\mathbf{e}})$, there is an eigenvalue λ_i such that

$$|\lambda_i - \hat{\lambda}| \leq \frac{\|(\mathbf{R} - \hat{\lambda}\Sigma)\hat{\mathbf{e}}\|_{\Sigma^{-1}}}{\|\Sigma\hat{\mathbf{e}}\|_{\Sigma^{-1}}},$$

where the norm $\|\mathbf{x}\|_{\Sigma^{-1}} \equiv (\mathbf{x}^T \Sigma^{-1} \mathbf{x})^{1/2}$. Computing this bound is made more difficult by this more complex norm; although the denominator is equal to $\hat{\mathbf{e}}^T \Sigma \hat{\mathbf{e}}$, to compute the numerator, it is necessary to compute the inverse of Σ and employ the Gohberg-Semencul relation.

Although the lack of a residual bound for generalized invariant subspaces makes verification of the accuracy of a computed subspace difficult, in practice, the computed generalized eigenvectors have been found to be of high quality as long as Σ is well conditioned. In cases where it is necessary to be assured of the accuracy of generalized eigendecompositions, it is possible to employ the $\mathcal{O}(M^2)$ look-ahead Levinson algorithm described in the previous chapter as the basis for the computation, ensuring that the results remain uncorrupted by instability.

CHAPTER 8

ESTIMATING THE NUMBER OF SIGNALS

To use the fast Toeplitz algorithms in subspace frequency estimation, it is necessary to know how many eigenvalues and eigenvectors must be computed; this is equivalent to knowing the number of signals present in the input data. In some cases, the number is known a priori, so that the fast algorithms may be used exactly as described in Chapter 5. In other cases, however, it is necessary to estimate the number of signals; this generally requires some additional computation beyond that needed when the number of signals is known.

As discussed Chapter 3, the most widely used estimators for the number of signals, the AIC (equations (3.10), (3.12)) and the MDL (equations (3.11), (3.13)), are of the form

$$h(\hat{\lambda}, n, M, L) = f(n, M, L) \log(\alpha(\hat{\lambda}, n, M, L)) + p(n, M, L),$$

where M is the dimension of $\hat{\mathbf{R}}$, $\hat{\lambda}$ is the set of eigenvalues of $\hat{\mathbf{R}}$, n is the hypothesized number of signals, and L is the number of points in the data record. The estimated number of signals \hat{N} is the value of n that minimizes $h(\hat{\lambda}, n, M, L)$. The function α in these estimators is the ratio of the geometric and arithmetic means of the $M - n$ smallest eigenvalues of $\hat{\mathbf{R}}$ (which must all be positive for this technique to be applied), and p is a penalty function that measures the complexity of the model. The function α is a measure of the inequality of the smallest eigenvalues; if all of the eigenvalues are equal, $\alpha = 1$, and since all of the eigenvalues are positive, $\alpha > 0$. Because all of the noise subspace eigenvalues of the exact autocorrelation matrix are equal (for

white noise), the nearness of α to one is related to the likelihood that the dimension of the noise subspace is $M - n$. The only part of h that depends on the autocorrelation estimate $\hat{\mathbf{R}}$ is

$$\alpha(\hat{\lambda}, n, M, L) = \frac{\left(\prod_{i=n+1}^M \hat{\lambda}_i \right)^{1/(M-n)}}{\frac{1}{M-n} \sum_{i=n+1}^M \hat{\lambda}_i},$$

When conventional methods are used to compute the eigenvalues, all of the $\hat{\lambda}_i$ are computed simultaneously, so that the value of h may then be computed for all n . Rather than computing each eigenvalue of $\hat{\mathbf{R}}$, fast Toeplitz eigendecomposition algorithms compute exact values for a subset of the $\hat{\lambda}_i$, and find bracket intervals that contain each of the remaining eigenvalues. These bracket intervals can be used to estimate the number of signals using conventional techniques such as the AIC and MDL.

Bounding the Number of Signals Estimate

Instead of calculating $h(\hat{\lambda}, n, M, L)$ exactly for each n , it is possible to use the eigenvalue brackets to find upper and lower bounds on the value of $h(\hat{\lambda}, n, M, L)$. If, for a certain value of n , say n' , it can be shown that the upper bound on $h(\hat{\lambda}, n', M, L)$ is less than the lower bound for all the other values of n , then $\hat{N} = n'$. In this way, the same estimated number of signals \hat{N} can be found, without explicitly computing all of the eigenvalues of $\hat{\mathbf{R}}$. To determine \hat{N} , then, we will need to compute upper and lower bounds on h for each n .

For any given n , the only quantity that is not known exactly is α , and because the logarithm is a strictly increasing function of its argument, it is easy to see that the minimum and maximum values of h will occur when α is at its upper or lower bound. Since $f(n, M, L) < 0$ for both the AIC and the MDL, the maximum value of

$h(\hat{\lambda}, n, M, L)$ for a given n will occur when $\alpha(\hat{\lambda}, n, M)$ is minimized, and the minimum when α is maximized. The problem has thus been reduced to determining bounds for α for each n . We will consider the problem in two parts: first, the computation of the upper and lower bounds for a given value of n , and second, efficient methods for computing bounds for $n + 1$ using information gained while computing the bounds for n .

Computing Bounds for a Single Value of n

The problem of minimizing h for a single value of n is a bound-constrained nonlinear optimization problem. Although the exact solution of this problem is generally too computationally expensive to be practical, it is possible to compute partial solutions to this problem that yield bounds for h .

The set of upper bounds for each of the $\hat{\lambda}$ will be denoted by the vector \mathbf{u} , and the lower bounds by the vector \mathbf{l} , so that

$$u_j \geq \hat{\lambda}_j \geq l_j.$$

As usual, the $\hat{\lambda}$ are ordered, so that $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_M$, and the upper and lower bounds \mathbf{u} and \mathbf{l} are ordered in the same way. To be certain that $\log(\alpha)$ is real valued, it is also necessary to require that $l_M > 0$, which corresponds to requiring that $\hat{\mathbf{R}}$ be positive definite.

To simplify the notation, we will denote the geometric mean of the last $M - n$ eigenvalues as

$$G(\hat{\lambda}, n, M) = \left(\prod_{i=n+1}^M \hat{\lambda}_i \right)^{1/(M-n)}$$

and their arithmetic mean as

$$A(\hat{\lambda}, n, M) = \frac{1}{M-n} \sum_{i=n+1}^M \hat{\lambda}_i.$$

Using this notation α is

$$\alpha(\hat{\lambda}, n, M) = \frac{G(\hat{\lambda}, n, M)}{A(\hat{\lambda}, n, M)},$$

and the derivative of α with respect to one of the eigenvalues $\hat{\lambda}_j$ is

$$\frac{\partial}{\partial \hat{\lambda}_j} \alpha(\hat{\lambda}, n, M) = \frac{1}{M-n} \alpha(\hat{\lambda}, n, M) \left(\frac{1}{\hat{\lambda}_j} - \frac{1}{A(\hat{\lambda}, n, M)} \right). \quad (8.1)$$

For a fixed n , the problem of minimizing or maximizing α exactly requires finding a vector $\hat{\lambda}$, each of whose elements lie within the bracket interval, that maximizes or minimizes $\alpha(\hat{\lambda}, n, M)$; the vectors that maximize and minimize α will be denoted $\hat{\lambda}^+(n)$ and $\hat{\lambda}^-(n)$, respectively. Some of the relations developed below apply to both the maximizing and minimizing vectors; for brevity, we will use the notation $\hat{\lambda}^\pm$ to indicate that an expression applies to either vector. (Of course, the substitution of $\hat{\lambda}^+$ or $\hat{\lambda}^-$ for $\hat{\lambda}^\pm$ in these expressions must be consistent.)

For each element $\hat{\lambda}_j^\pm(n)$ of either the maximizing vector $\hat{\lambda}^+(n)$ or the minimizing vector $\hat{\lambda}^-(n)$, one of the following three mutually exclusive conditions is true [92, p. 77]:

- A: $\hat{\lambda}_j^\pm(n) = u_j$;
- B: $\hat{\lambda}_j^\pm(n) = l_j$;
- C: $u_j > \hat{\lambda}_j^\pm(n) > l_j$ and $\frac{\partial}{\partial \hat{\lambda}_j} \alpha(\hat{\lambda}^\pm, n, M) = 0$.

If we examine equation (8.1), we see that, because $\alpha/(M-n)$ must be positive, the sign of the derivative with respect to $\hat{\lambda}_j$ is determined by the relative size of

$\hat{\lambda}_j^\pm$ and $A(\hat{\lambda}^\pm, n, M)$. The derivative will be positive if $\hat{\lambda}_j^\pm < A(\hat{\lambda}^\pm, n, M)$, zero if $\hat{\lambda}_j^\pm = A(\hat{\lambda}^\pm, n, M)$, and negative if $\hat{\lambda}_j^\pm > A(\hat{\lambda}^\pm, n, M)$. The value of $A(\hat{\lambda}^\pm, n, M)$ is not known exactly, but it is bounded by $A(\mathbf{u}, n, M)$ and $A(\mathbf{l}, n, M)$, and this may enable us to determine some of the elements of $\hat{\lambda}^+$ and $\hat{\lambda}^-$.

If the bracket intervals for an eigenvalue $u_j \geq \hat{\lambda}_j^\pm \geq l_j$ and those for the arithmetic mean $A(\mathbf{u}, n, M) \geq A(\hat{\lambda}^\pm, n, M) \geq A(\mathbf{l}, n, M)$ do not overlap, then $\frac{\partial \alpha}{\partial \hat{\lambda}_j}$ cannot be zero for any $\hat{\lambda}_j$ within the bracket, and it is possible to determine the values of $\hat{\lambda}_j^-$ and $\hat{\lambda}_j^+$ easily. If the partial derivative of α with respect to $\hat{\lambda}_j$ is positive for all $\hat{\lambda}_j$ within the bracket interval, then $\hat{\lambda}_j^- = l_j$, and $\hat{\lambda}_j^+ = u_j$; if the derivative is negative over the entire interval, then $\hat{\lambda}_j^- = u_j$, and $\hat{\lambda}_j^+ = l_j$.

As each element of $\hat{\lambda}^+$ and $\hat{\lambda}^-$ is determined in this fashion, the bracket intervals for $A(\hat{\lambda}^+, n, M)$ and $A(\hat{\lambda}^-, n, M)$ are narrowed, possibly allowing other elements of the minimizing and maximizing vectors to be determined. Since the global maximum of α is one, and occurs when all of the $\hat{\lambda}$ are equal, we can see that maximizing α requires making the $\hat{\lambda}$ as close to equal as possible (that is, as close to the arithmetic mean as possible), and minimizing it requires that the $\hat{\lambda}$ be as unequal (as far from the arithmetic mean) as possible. The iterative determination of components of $\hat{\lambda}^+$ and $\hat{\lambda}^-$, and computation of an upper bound $\alpha^+(n)$ on $\alpha(\hat{\lambda}, n, M)$ is performed by Algorithm 8.1 below. The computation of a lower bound is performed in exactly the same fashion.

ALGORITHM 8.1: BOUNDING α

```

start = M - n + 1
finish = M
changed = true
while changed
  S0 = sum(l(M - n + 1 : start - 1))
  S1 = sum(l(start : finish))
  S2 = sum(min(u(start : finish), l(start - 1)))
  S3 = sum(u(finish + 1 : M))
  A+ = (S0 + S2 + S3)/(M - n)
  A- = (S0 + S1 + S3)/(M - n)
  changed = false
  for i = start : finish
    if li ≥ A+
      start = i + 1
      changed = true
    else if ui ≤ A-
      finish = i - 1
      changed = true
    end
  end
end
G+ = (prod(l(M - n + 1 : start - 1)) * prod(min(u(start : M), l(start - 1))))1/(M-n)
α+(n) = min(1, G+/A-)

```

After determining as many of the components of $\hat{\lambda}^+$ as possible, Algorithm 8.1 computes the maximum value of α by taking the ratio of the largest possible geometric mean to the smallest possible arithmetic mean (or the reverse, when the minimum of α is being calculated). This is a suboptimal choice, since the values assumed for the unknown eigenvalues are different in the computation of the two means. Finding a consistent maximizing or minimizing vector requires much more computation, but if the number of remaining eigenvalues is small enough, it may be feasible to solve the exact bound-constrained minimization or maximization problem. This approach is only viable when the number of elements whose values are not known is small (5–20).

Narrowing the Bracket Intervals

When the bracket intervals resulting from the computation of a small number of eigenvalues are used in the above procedure, the resulting bounds on h are often too loose to allow the number of signals estimate to be determined. This problem is particularly common when a signal subspace frequency estimator is being used, because these estimators compute the largest eigenvalues of $\hat{\mathbf{R}}$, first, while the number of signals estimators employ the smallest eigenvalues. It is of course possible to continue to compute eigenvalues until a large number are known exactly, and the bounds on those remaining are narrow enough that the number of signals estimate may be determined. Generally, more than half of the eigenvalues must be computed before this occurs, and the cost per eigenvalue is approximately 8 solutions of the Yule-Walker equation.

In many cases, the number of signals may be estimated with many fewer solutions by employing the spectrum slicing technique described in Chapter 6 at selected values of λ . Assume that the eigenvalues that have not yet been computed exactly lie in the range $\lambda_U \geq \lambda_i \geq \lambda_L$. By slicing the spectrum within this range at intervals of δ , the bracket interval for every unknown eigenvalue can be reduced to no more than δ . The cost for this is $(\lambda_U - \lambda_L)/\delta$ slices. In contrast to computing additional eigenvalues, each slice costs only one Yule-Walker solution, and the new information obtained is spread across the entire region, rather than being concentrated near a certain group of eigenvalues.

When this technique is employed, it is often possible to determine the number of signals estimate with 20–50 additional Yule-Walker solution, equivalent to the computation of 3–6 additional eigenvalues. In addition, it is possible to subdivide the interval coarsely, check whether the number of signals estimate can be determined, and divide the interval more finely only if necessary.

Computing Bounds for Consecutive Values of n

Once a set of brackets for $\hat{\lambda}^+(n)$ and $\hat{\lambda}^-(n)$ has been determined, it is possible to bracket $\hat{\lambda}^+(n+1)$ and $\hat{\lambda}^-(n+1)$ with less computation. The bracket computation for n produces a pair of indices *start* and *finish* that indicate the region within which the elements of the minimizing or maximizing vector are not exactly known. This region is occupied by those components of $\hat{\lambda}^\pm$ that are within the range of possible values for the mean $A(\hat{\lambda}^\pm, n, M)$.

Because the $\hat{\lambda}$ are in decreasing order, $A(\hat{\lambda}^\pm, n+1, M) \leq A(\hat{\lambda}^\pm, n, M)$; this means that the values of *start* and *finish* cannot increase. The additional computation required for $n+1$ is simply the examination of the vectors near these boundaries; if the range of possible values for the arithmetic mean has moved downward so that it no longer overlaps an eigenvalue bracket, then *start* has decreased, and the value of some elements of the minimizing or maximizing vector may now be found exactly. Similarly, if the range of values for the arithmetic mean has moved downward to overlap a bracket interval, than that element's value can no longer be determined exactly.

Although the principal focus of this work is on the subspace frequency estimators themselves, rather than the number of signals estimation problem, this chapter has outlined a simple technique for obtaining estimates of the number of signals when the fast Toeplitz eigensolvers are used. Unfortunately, unlike the standard implementations of the subspace techniques, the fast Toeplitz implementations usually require additional computation if the number of signals is not known a priori. The possibility of more elegant solution to this problem is a subject for future investigation.

CHAPTER 9

COMPARISON OF CONVENTIONAL AND FAST METHODS

In this chapter, the fast Toeplitz eigendecomposition methods are used in an implementation of the ESPRIT method for frequency estimation. If a Toeplitz autocorrelation estimate is used, the fast implementation provides a much more efficient method of computing subspace frequency estimates. As we saw in Chapter 4, when a Toeplitz matrix is used in place of a covariance matrix of the same size, some performance degradation occurs. Because of the superior efficiency of the fast Toeplitz algorithms, however, it is often possible to use a larger matrix and still reduce the computation time. We will see below that in many cases, the improvement in the accuracy of the frequency estimate due to the use of a larger $\hat{\mathbf{R}}$ more than offsets the degradation due to the use of a Toeplitz estimate. The result is that more accurate frequency estimates may be obtained in less time, when compared to conventional implementations of the subspace methods.

The analysis in Chapter 4 also showed that the relative performance of the various estimators depended on several characteristics of the input data, including the number of samples in the data record, the signal-to-noise ratio, and the frequency difference between the sinusoids in the input data. To provide a clear illustration of the effects of these changes, we will consider cases in which the effect of each of these factors is clearly seen, examining the variations in the accuracy and execution time of the conventional and fast methods for subspace frequency estimation. First, however, a detailed description of the implementation of each of these approaches will be given.

Implementation of the ESPRIT Algorithm

Two different forms of the TLS-ESPRIT algorithm described in Chapter 3 have been implemented. The first uses the most efficient conventional techniques to compute the eigendecomposition, and may be used with either Toeplitz or covariance estimators of the autocorrelation matrix; the second uses the fast Toeplitz eigensolver described in Chapter 5. Other than the method of computing the invariant subspace, the two variants are identical.

Eigendecomposition by the Conventional Method

The conventional variant calculates the eigenvalues and eigenvectors by standard methods, using routines from the linear algebra packages LINPACK [93] and EISPACK [94, 95]. These methods make no assumption about the structure of the matrix, except for symmetry; they may be used with any of the autocorrelation matrix estimators discussed here.

In cases where only a few eigenvectors are required, $\hat{\mathbf{R}}$ is reduced to tridiagonal form using the EISPACK routine `tred1`, and its eigenvalues are calculated using QL iteration by the routine `tql1`. Reduction of an $M \times M$ matrix to tridiagonal form requires approximately $4M^3/3$ operations; the cost of computing the eigenvalues of the resulting tridiagonal matrix is negligible by comparison. Once the eigenvalues have been found, the eigenvectors are calculated by inverse iteration using the LINPACK routines `dsifa` and `dsisl`, which solve symmetric indefinite systems. Each iteration requires $M^3/3$ operations, and since the eigenvalues are very accurate, only one iteration is usually needed to produce an accurate eigenvector. The total cost to produce N eigenvalues and their associated eigenvectors is approximately $(N + 4)M^3/3$ operations.

If many eigenvectors are required, the EISPACK routines `tred2` and `tql2` are used instead. These also perform Householder reduction to tridiagonal form and

tridiagonal QL iteration, but they accumulate the orthogonal transformations used to perform these steps, so that when the process is complete, all the eigenvectors and eigenvalues of $\hat{\mathbf{R}}$ are available. The total computational cost is approximately $9M^3$ operations, so if more than 23 eigenvectors are needed, it is more efficient to use these routines to compute them all, rather than use inverse iteration.

Eigendecomposition by the Fast Method

The fast variant computes eigenvalues and eigenvectors of a Toeplitz autocorrelation matrix estimate using the fast and superfast techniques described in Chapter 5. The crossover point between the fast and superfast Toeplitz solvers was chosen to be $M = 513$, since the superfast techniques become more efficient at this point. Thus computation of N eigenvalues and eigenvectors for $M < 513$ requires $\mathcal{O}(NM^2)$ operations, and $\mathcal{O}(NM \log^2 M)$ operations are needed for $M \geq 513$. As described in Chapter 5, the accuracy of the computed eigenvalues and eigenvectors is controlled by the tolerance ϵ , which is the precision to which a root of $E(\lambda)$ is located.

The number of Toeplitz solutions necessary to locate an eigenvalue depends on the tolerance ϵ , as well as on the number of nearby eigenvalues previously computed, since information from nearby solutions provides the root finder with a better starting point. For the computations described here, $\epsilon = 10^{-12}$ was used, and the number of Toeplitz solutions required to locate the first eigenvalue was approximately 10–12. Subsequent nearby eigenvalues required fewer evaluations; the average number of solutions required for a cluster of contiguous eigenvalues was between 8 and 10. To compute each solution, the fast and superfast Toeplitz solvers require approximately $2M^2$ and $8M \log_2 M$ operations, respectively.

Because the accuracy of ESPRIT frequency estimates is entirely determined by the accuracy of the computed invariant subspace, a single inverse iteration is used to improve the accuracy of each eigenvector after the associated eigenvalue is

found. For $M < 64$, Trench's algorithm is used to solve the resulting (indefinite) Toeplitz system; for $M \geq 64$, it is more efficient to calculate the solution to the Yule-Walker problem using a fast or superfast Toeplitz solver, and use the Gohberg-Semencul relation to compute the solution. Since the inverse iteration requires a single additional solution of the Yule-Walker problem, the total cost for locating a cluster of N eigenvalues and their associated eigenvectors is approximately $20NM^2$ for $M < 513$, and $80NM \log_2 M$ for $M \geq 513$.

Although the residual computations described in Chapter 7 were performed on each of the calculated eigenvalues and eigenvectors, with the threshold on the angle between the exact and computed eigenvector set at 10^{-3} , no eigenvalues were rejected in any of the trials performed here. In addition, empirical tests showed almost no difference between the frequency estimates computed with fast techniques and those computed using stable methods. If the same Toeplitz matrix was used to compute frequency estimates with the conventional (stable) methods and the fast methods, it was not unusual for the resulting frequency estimates to agree to 15 digits, and the largest difference in a frequency estimate observed in several thousand trials was in the 11th digit. This indicates that the fast techniques give very good estimates of the invariant subspaces, and that the residual bounds need only be used to check for the extremely rare cases where numerical instability causes large errors.

The ESPRIT Computation

Once the desired invariant subspace has been computed, the conventional and fast implementations perform exactly the same computational steps. The TLS solution to the ESPRIT equation $\mathbf{E}_1 \mathbf{X} = \mathbf{E}_2$ is computed using the partial TLS algorithm developed by Van Huffel [96]; this algorithm computes only the portion of the singular value decomposition necessary to solve the TLS problem, and requires approximately

the same number of operations as the standard TLS solution method using the R-SVD [37, p. 577]. This step requires about $2MN^2$ operations to compute \mathbf{X} .

The eigenvalues of the TLS solution \mathbf{X} are computed by reducing it to upper Hessenburg form using the EISPACK routine `elmhes`, and computing the eigenvalues using the routine `hqr`. The total computation required by this step is approximately $10N^3$ operations. The TLS-ESPRIT frequency estimates are the arguments of the eigenvalues of \mathbf{X} .

Relative Complexity

By summing the computational costs given in the section above, we can see that the cost of estimating the frequencies of N complex signals using the TLS-ESPRIT algorithm is approximately $2MN^2 + 10N^3$ operations, plus the cost of computing the desired invariant subspace. Computation of the subspace requires approximately $(N + 4)M^3/3$ operations for the conventional methods, $20NM^2$ for the fast methods, and $80NM \log_2 M$ for the superfast methods.

The worst case for the fast and superfast algorithms is when all of the eigenvalues are computed, that is, $N = M$. In this case, the fast algorithms are always more complex than conventional methods. It is interesting to note, however, that the superfast techniques are asymptotically less complex than the conventional methods even if all of the eigenvalues and eigenvectors are computed. Because the conventional methods require about $9M^3$ operations, the superfast methods are less complex than conventional techniques for $M > 1025$. In practice, the better locality of reference of the conventional techniques renders them slower until $M \approx 2049$. Although subspace frequency estimation rarely requires such large numbers of eigenvectors, the superfast methods are the first Toeplitz eigendecomposition algorithms for which the asymptotic complexity is lower than the standard methods even when all eigenvalues and eigenvectors are computed.

Simulation Results

In this section, the results of the fast and conventional variants are compared for a variety of different input signals, and the effects of variations in data record length, signal-to-noise ratio, and frequency spacing are examined. Both the accuracy of the estimates, in terms of the mean squared error and variance, and the efficiency of the computation, in terms of the total computation time, are compared. The cases considered are summarized in Table 9.1 below, which also lists the figures in which the relevant results are shown.

Table 9.1: Summary of Test Cases

Case	Results
Low signal-to-noise ratio	Figures 9.1 – 9.3
Long data record	Figures 9.4–9.5
High signal-to-noise ratio	Figures 9.6–9.9
Closely spaced signals	Figures 9.10–9.12

The accuracy of the techniques is compared in terms of the statistics of the frequency estimates computed for an ensemble of input signals with identical sinusoidal components, but different pseudorandom Gaussian noise. To determine these statistics, each problem was solved 100 times using each of the two variants. The pseudorandom number generator that produced the noise in each input data record was reinitialized to the same seed value for each variant; this ensured that the input data were identical for the fast and conventional versions.

Low Signal-to-Noise Ratio

The input for the first case is a data record with $l = 1000$ samples, composed of two sinusoids with parameters $a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 2.01062$, $\phi_2 = -0.4$, in additive white Gaussian noise with $\sigma^2 = 1.0$. This represents a relatively low signal-to-noise ratio of -3 dB, and is one of the cases for which we

expect the fast Toeplitz methods to have good performance compared to the standard variants. The mean squared error in the frequency estimate $\hat{\omega}_1$ is shown in Figure 9.1. As before, the Bhattacharyya bound will be shown for comparison on graphs of both mean squared error and variance, even though it only bounds the variance.

Even for this relatively small data record length, the performance of the fast Toeplitz method is competitive with the conventional technique, with an increase in the mean squared error due to the use of a Toeplitz autocorrelation estimate of less than a factor of three. As was the case in Chapter 4, the additional error is almost entirely due to bias, as may be seen from the comparison of the variances in Figure 9.2.

Of course, the reason for employing a Toeplitz autocorrelation estimate is to reduce the time required to estimate the frequency. When the performance of the two techniques is compared on this basis, we can see that the fast method shows a small advantage for the larger autocorrelation matrices, and is competitive even for the smallest matrices considered, for which $M = 33$. The performance as a function of computation time is shown in Figure 9.3.

Long Data Records

The accuracy of the fast Toeplitz techniques approaches that of the conventional variants as the number of samples in the data record becomes large. In addition, the efficiency advantage of the fast variants becomes much larger as M increases. We would expect, then, that the fast variants would show the greatest advantage over the conventional implementations when long data records are used. Figure 9.4 shows the mean squared error in the frequency estimates for a such a case, where the signal parameters are $a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 2.01062$, $\phi_2 = -0.4$, and $\sigma^2 = 100$. The record length L is 40000 samples.

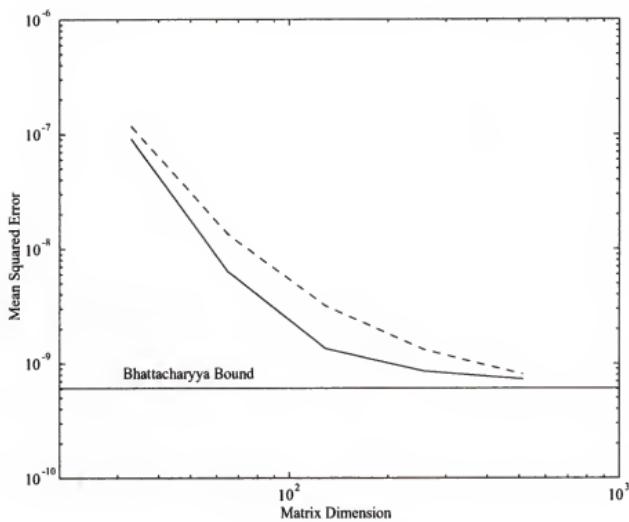


Figure 9.1: Mean Squared Error in $\hat{\omega}_1$ versus Matrix Dimension M for the Conventional (solid line) and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the low SNR case

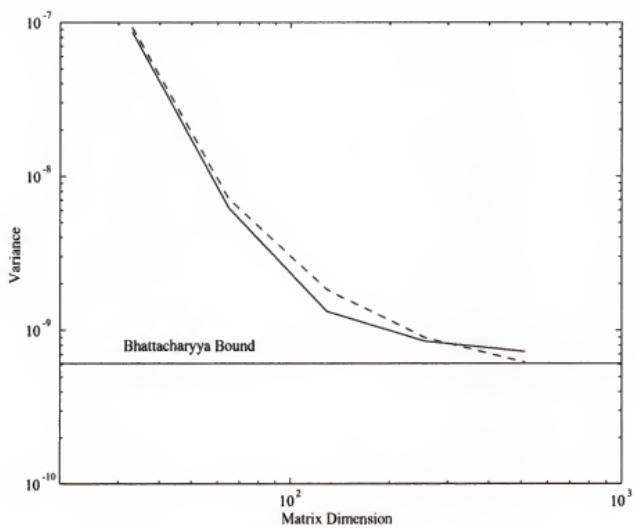


Figure 9.2: Variance in $\hat{\omega}_1$ versus Matrix Dimension M for the Conventional (solid line) and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the low SNR case

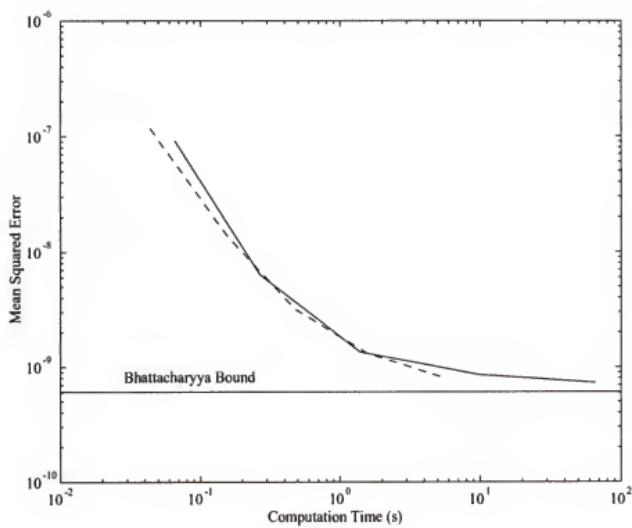


Figure 9.3: Mean Squared Error in $\hat{\omega}_1$ versus Computation Time for the Conventional (solid line) and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the low SNR case

Because the number of samples is large, the Toeplitz estimate of the autocorrelation is very close to the covariance estimate; this represents a very favorable case for the fast Toeplitz techniques. (The signal-to-noise ratio is also low, but this has a much smaller effect than the data record length.) The mean squared errors in the estimates produced by the fast and conventional variants are virtually identical, as seen in Figure 9.4; the mean difference is less than 5%. (Because conventional methods require a large amount of computation, the conventional estimates were computed only up to $M = 1025$, while the fast variants were computed up to $M = 16385$.) Both estimates have negligible bias, and due to the low signal-to-noise ratio, neither approaches the Bhattacharyya bound.

The efficiency advantage of the fast variants becomes compelling for $M \geq 256$. For large data records, the fast variants make it possible to compute eigendecompositions of matrices which are far larger than the largest practical with conventional techniques, and so the performance disadvantage of using a Toeplitz estimate may be overcome by increasing M . Figure 9.5 shows the mean squared error as a function of execution time, where the dramatic advantage of the fast techniques is apparent: for the same execution time, the mean squared error of the fast Toeplitz estimate is as much as two orders of magnitude less than that of the conventional variant.

The computation time for the fast variant with $M = 16385$ was approximately the same as that of the conventional estimate for $M = 1025$; computing a conventional estimate for $M = 16385$ would have required four thousand times more computation. For the system used to compute these examples, a workstation capable of approximately 8 million floating-point operations per second, the fast Toeplitz estimate with $M = 16385$ and the conventional estimate with $M = 1025$ both required approximately 10 minutes. Computing a conventional estimate with $M = 16385$ would have required four weeks, and 1 gigabyte of memory would have been needed for packed storage of the covariance estimate.

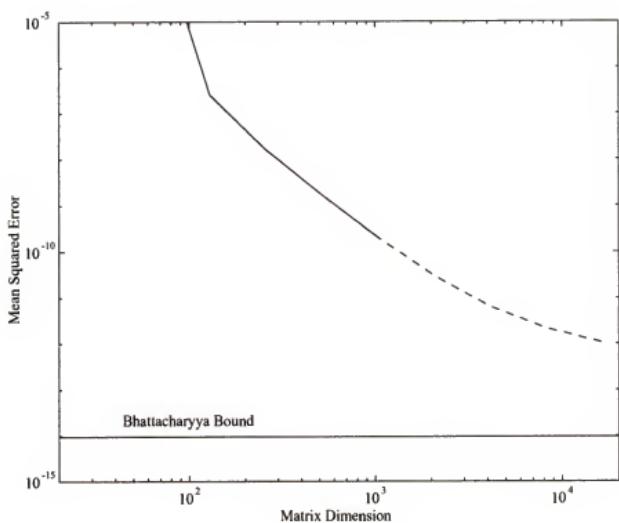


Figure 9.4: Mean Squared Error in $\hat{\omega}_1$ versus Matrix Dimension M for the Conventional (solid line) and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for $L = 40000$

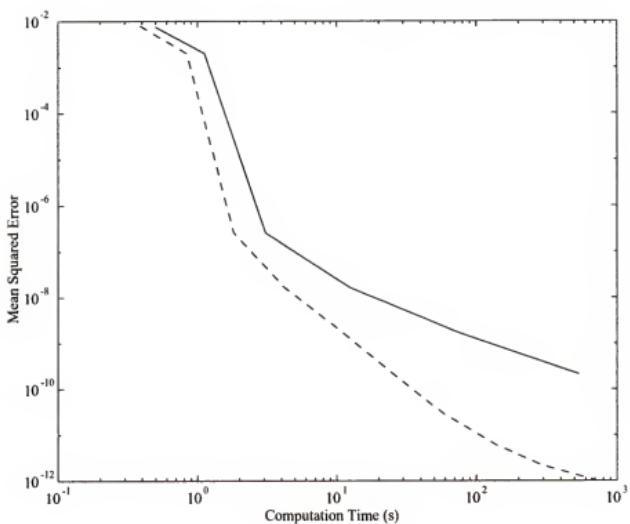


Figure 9.5: Mean Squared Error in $\hat{\omega}_1$ versus Computation Time for the Conventional (solid line) and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for $L = 40000$

The large data record case for which the fast variant shows the greatest advantage is not the only case in which subspace estimates are used, but it is certainly not uncommon. For example, the 40000 point data record used in the above example would be less than one second of CD-rate digital audio. In general, if we examine the situations in which subspace estimators are used, we can see that there are two factors which limit the size of the autocorrelation matrix estimate. The first is the length of the input data record, since M must be no greater than L ; the second is the time available for computation. If M must be small because the data record is short, the fast Toeplitz approach has little advantage, although its performance may still be quite close to the conventional variant. If M is limited by the computation required, however, the fast techniques can improve the accuracy of the frequency estimate dramatically, as shown by the preceding example.

A Less Favorable Case

The two examples above show instances in which the performance of the fast variant is equal to or superior to the conventional version. To form a balanced picture of the relative performance of the two approaches, it is also interesting to examine a case where the fast variants are inferior. This case has signal parameters $a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 2.01062$, $\phi_2 = -0.4$, $\sigma^2 = 0.01$, and a record length $L = 2000$, representing a high signal-to-noise ratio and a relatively short data record. In contrast to the earlier cases, this is close to the worst case for the fast Toeplitz methods. The mean squared error is shown in Figure 9.6. (In the two previous cases, the choice of Toeplitz estimator had very little effect on the results; since the choice of estimator has a larger effect in this and the following cases, the results of using both the biased and unbiased Toeplitz autocorrelation estimates are shown.)

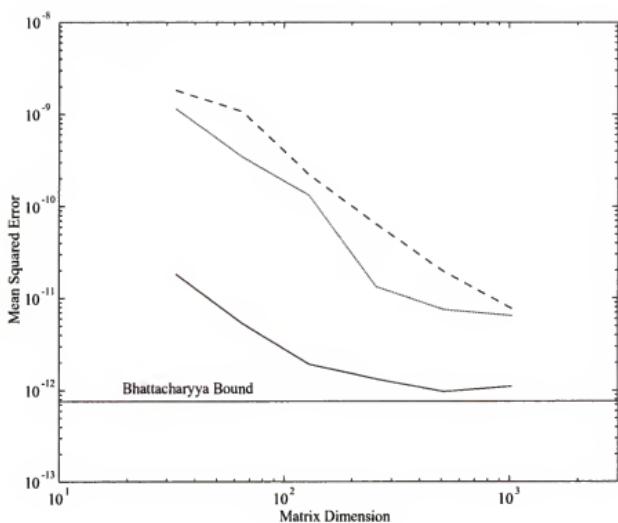


Figure 9.6: Mean Squared Error in $\hat{\omega}_1$ versus Matrix Dimension for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the high SNR, short data record case

Although the mean squared error in the fast estimates is at least a factor of ten larger than the conventional estimate, almost all of the increase in error for the fast techniques is due to the increased bias inherent in the use of a Toeplitz matrix. When the techniques are compared on the basis of variance, the advantage of the conventional approach is much smaller, and actually disappears for large M , as seen in Figure 9.7.

As before, when the two variants are compared on the basis of execution time, the fast variants fare better. Figure 9.8 shows the mean squared error versus computation time; comparison on this basis reduces the advantage of the conventional method by a factor of approximately 1.5–2. Because the variances of the two approaches are very similar, the fast variants actually have an advantage when the variance is compared on the basis of execution time. This comparison is shown in Figure 9.9.

Even for the unfavorable situation considered in this example, the fast variants are competitive with the conventional implementations when variance of the frequency estimate, rather than mean squared error, is the primary concern. This is consistent with the behavior of the subspace techniques with Toeplitz autocorrelation estimates seen in Chapter 4.

Closely Spaced Signals

Finally, we will examine the performance of the two variants on a signal with two very closely spaced sinusoids. The input signal for this case has $a_1 = 1$, $\omega_1 = 1.88496$, $\phi_1 = 0.3$, $a_2 = 1$, $\omega_2 = 1.88621$, $\phi_2 = -0.4$, with $\sigma^2 = 0.0001$ and $L = 2000$. The frequency spacing between the two signals is $\Delta f = 0.0002$, which is a factor of 2.5 smaller than the classical resolution limit for $L = 2000$. The very low noise level is necessary for any of the estimators to resolve the signals; even the conventional

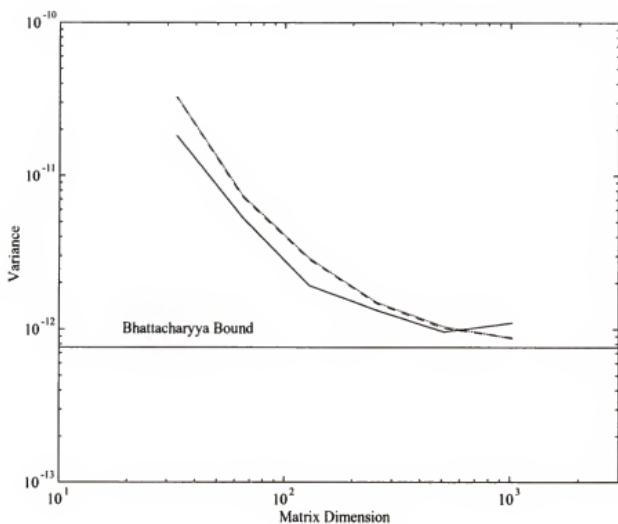


Figure 9.7: Variance in $\hat{\omega}_1$ versus Matrix Dimension for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the high SNR, short data record case

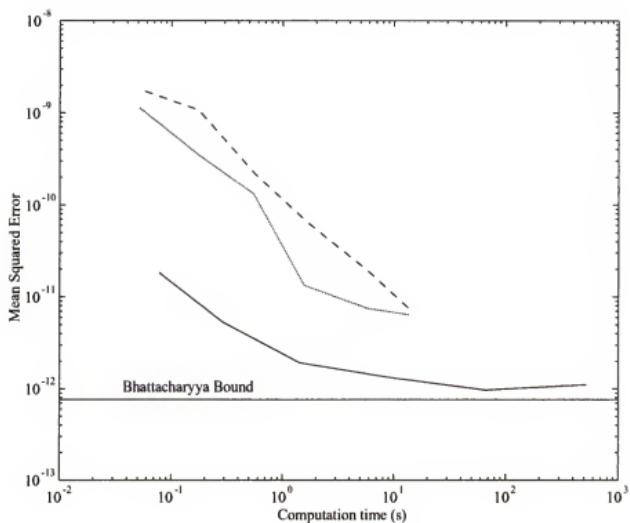


Figure 9.8: Mean Squared Error in $\hat{\omega}_1$ versus Computation Time for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the high SNR, short data record case

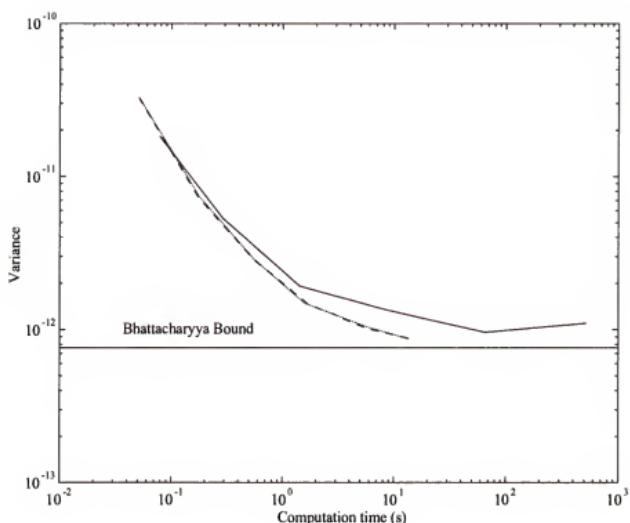


Figure 9.9: Variance in $\hat{\omega}_1$ versus Computation Time for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for the high SNR, short data record case

technique does not resolve them for higher noise powers. The mean squared error as a function of matrix dimension is shown in Figure 9.10.

As we saw in Chapter 4, the use of a Toeplitz estimate reduces the resolution of the subspace techniques somewhat, although they are still capable of resolving signals closer than the classical resolution limit. The behavior of estimates computed using the two Toeplitz estimators is also similar to that seen earlier; it is necessary for M to be approximately $L/2$ before the unbiased estimator resolves the signals, at which point it abruptly transitions to much higher accuracy. The biased Toeplitz estimator resolves the signals at all values of M , and in fact has lower variance than the covariance estimator, as seen in Figure 9.11.

As would be expected, the fast techniques compare better with the conventional variant when computation time is used as the basis for comparison. The variance of the frequency estimate versus computation time is shown in Figure 9.12.

The frequency estimation accuracy of the fast Toeplitz ESPRIT algorithm is extremely similar to the accuracy of Toeplitz-based estimators examined in Chapter 4; it appears that the only losses in performance are those due to the use of a Toeplitz estimate. No loss of accuracy due to numerical instability of the fast or superfast algorithms for Toeplitz eigendecomposition has been observed, and the residual tests described in Chapter 7 verify that loss of accuracy due to numerical instability is an extremely rare event.

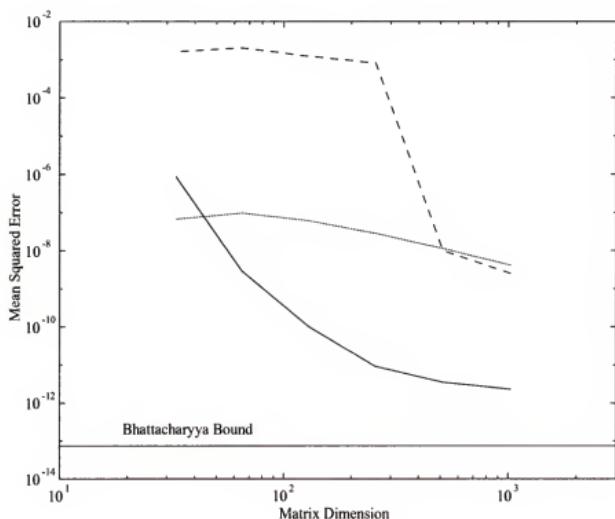


Figure 9.10: Mean Squared Error in $\hat{\omega}_1$ versus Matrix Dimension for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for $L = 2000$ and $\Delta f = 0.0002$

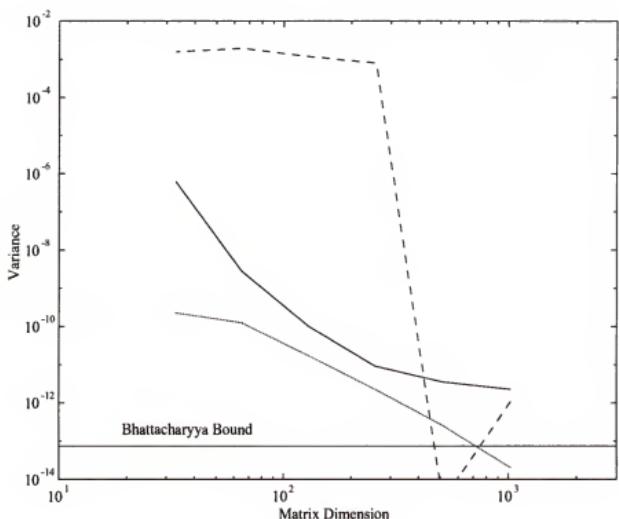


Figure 9.11: Variance in $\hat{\omega}_1$ versus Matrix Dimension for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for $L = 2000$ and $\Delta f = 0.0002$

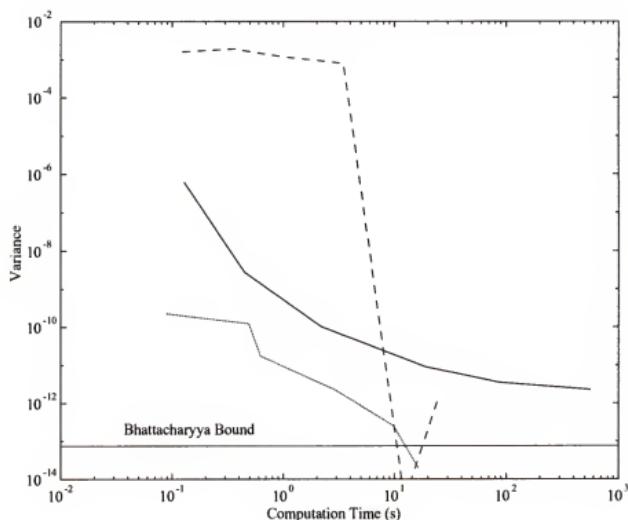


Figure 9.12: Variance in $\hat{\omega}_1$ versus Computation Time for the Conventional (solid line), Biased Fast Toeplitz (dotted line), and Unbiased Fast Toeplitz (dashed line) ESPRIT Methods, for $L = 2000$ and $\Delta f = 0.0002$

CHAPTER 10 CONCLUSIONS

The preceding chapters have attempted to present a balanced assessment of the fast Toeplitz techniques for frequency estimation. We have seen that, when a large input data record is available, these new variants of the subspace methods can greatly reduce the computation time required for subspace frequency estimation. It is also possible in many circumstances to obtain a more accurate estimate in the same amount of time, by using the efficiency improvement from the fast algorithm to increase the size of the autocorrelation matrix, rather than reduce the computation time.

Although these techniques appear quite promising, it is important to keep in mind that the utility of any new method can be only established by applying it to a variety of actual problems; assessing the promise of a technique on the basis of its performance in situations designed to maximize its advantages and minimize disadvantages can only give a distorted view. To ensure that the possible shortcomings of this method are fully discussed, a review of the method from a critical perspective is presented below.

A Critical Review

The approach described here dramatically reduces the computation required to compute eigenvalues and eigenvectors of large Toeplitz matrices, making it possible to compute subspace frequency estimates much more rapidly. The use of a Toeplitz estimate of the autocorrelation is essential to the efficiency of this technique, however, it is also the cause of its most important limitations.

The most serious difficulty with this method is that, for short or moderate data record lengths, the use of a Toeplitz estimate degrades the performance of the subspace frequency estimators somewhat. The degradation includes an increase in the bias of the frequency estimates, and, for small autocorrelation matrices, a reduction of the ability to resolve very closely spaced signals. In order to preserve the high resolution of the subspace techniques, it is necessary either to accept the relatively high bias associated with the use of the biased estimator, or to use the unbiased estimator with a matrix dimension of at least half the length of the data record.

The loss of performance, in comparison with the standard subspace methods, is most severe when the signal-to-noise ratio is high and the data records are short. In many circumstances, it is possible to make up for this degradation by using a larger autocorrelation matrix, and when computation time, rather than data record length, is the limiting factor, the fast methods can give much better accuracy than conventional methods. However, the results would be much more satisfying if the performance penalty could be eliminated completely, by developing a fast method which can be used with covariance estimates. Two possible approaches for doing this are outlined below. Because of the performance loss inherent in the use of a Toeplitz estimator, the current technique is attractive principally in cases where long data records are available, so that the performance of the Toeplitz algorithms are close to the standard versions.

Another limitation of this method concerns its application to a slightly different problem: in many cases, identical algorithms may be used for frequency estimation and for the closely related problem of array bearing estimation. Although this is also true for the algorithms presented here, there are several restrictions. First, the requirement that a Toeplitz autocorrelation matrix be used limits the applicability of this technique to uniformly spaced linear arrays. The standard versions of the MUSIC and ESPRIT estimators are useful for a much broader range of array geometries.

A more serious limitation arises from purely practical considerations. In the frequency estimation problem, the upper limit to the size of the autocorrelation matrix is set by the length of the data record; in the bearing estimation problem, the limit is set by the size of the array. While data records with thousands of samples are extremely common, uniform linear arrays with thousands of elements are rare. There are very few real arrays in existence large enough for these algorithms to offer a significant advantage, and while synthetic aperture techniques can easily produce arrays with very large numbers of synthesized elements, synthetic aperture systems are principally used for imaging distributed targets, not for the point target bearing estimation problem that the subspace algorithms solve.

Areas for Further Research

The work described here touches on many fields of current research, and has led to several promising avenues which will bear further investigation. A brief summary of several of the most important is given in the sections below.

Parallel Implementations

A very important aspect of the procedures discussed in this work is their inherent parallelism. The computation of eigenvalues and eigenvectors proceeds in two phases: calculation of bracket intervals, and location of eigenvalues; each of these phases is easily performed in parallel. The bracket computation requires solution of the Yule-Walker equation defined by $\mathbf{T} - \lambda \mathbf{I}$ over a range of values of λ . By dividing the range of values for λ into subranges, and assigning each subrange to a different processor, the bracket computation can be performed in parallel. The interprocessor communication required is very small, since the vector which defines \mathbf{T} can be broadcast to all processors on initialization, and the only further information needed by each processor is a range of λ in which to operate.

Once bracket intervals for the desired eigenvalues have been established, the computation of each eigenvalue and the associated eigenvector can also proceed in parallel, with one processor assigned to each eigenvalue. If at least N processors are available, then the computation of N eigenvalues and eigenvectors of an $M \times M$ Toeplitz matrix can be performed in $\mathcal{O}(M \log^2 M)$ time. In addition, each computation is completely independent, so no interprocessor communication is required.

In contrast to standard methods for parallel eigenvalue computation, such as the parallel Jacobi method [37, 97], eigenvalue and eigenvector computation with the fast Toeplitz methods is very coarse-grained, and requires little interprocessor communication. These characteristics are a great practical advantage, because they allow the algorithm to be efficiently implemented on a wide range of common multiprocessor systems, including both SIMD and MIMD systems, and make its efficiency relatively insensitive to the details of interprocessor communication on a particular system. Furthermore, considering the advantages offered by DSP processors for computing the FFTs used in the superfast techniques, and the recent appearance of DSP processors designed for parallel applications, it is clear that the methods described here are promising candidates for parallel processing systems for frequency estimation.

Maximum-Likelihood Toeplitz Estimation

In addition to the well-known methods of autocorrelation estimation described in Chapter 4, a family of new methods has recently been developed. The covariance method produces the maximum likelihood estimate of the autocorrelation matrix under the stochastic signal model, when no constraints are imposed on the structure of \mathbf{R} . However, the autocorrelation of any signal consistent with the stochastic signal model is known to be Toeplitz, and this fact can presumably be used to produce a more accurate estimate. Methods for computing the maximum likelihood Toeplitz estimate of the autocorrelation were first discussed by Burg, Luneberger, and Wegner [98],

whose paper has sparked several other workers to consider the problem of constrained maximum likelihood estimation of \mathbf{R} [99, 100].

The maximum likelihood estimate is found by maximizing the function

$$f(\hat{\mathbf{R}}_T) = -\log(\det(\hat{\mathbf{R}}_T)) - \text{trace}(\hat{\mathbf{R}}_T^{-1}\hat{\mathbf{R}}_C),$$

where $\hat{\mathbf{R}}_T$ is a Toeplitz estimate, and $\hat{\mathbf{R}}_C$ is the covariance estimate. The maximum likelihood estimate is the value of $\hat{\mathbf{R}}_T$ which maximizes f . Notice that this maximization requires computing the determinant and inverse of a Toeplitz matrix, and the product of the inverse of a Toeplitz matrix with $\hat{\mathbf{R}}_C$; efficient algorithms for these operations were given in Chapter 5. Using these algorithms, f can be evaluated in $\mathcal{O}(M^2 \log M)$ operations.

This work has not attempted a comprehensive study of the performance of the Toeplitz maximum likelihood method for estimation of sinusoidal frequencies. However, it should be noted that, in contrast to the very promising results reported by Williams [100] for the array bearing estimation problem, a series of simulations for the frequency estimation problem show that the maximum likelihood Toeplitz estimator has performance similar to the unbiased Toeplitz estimator, which is much simpler to compute. Nevertheless, the fast techniques described in Chapter 5 may be useful for improving the efficiency of calculating the maximum likelihood Toeplitz estimator in other contexts.

Displacement Rank Algorithms for Covariance Matrices

A group of workers associated with Kailath have developed algorithms for the solution of matrix equations in which the matrix has low displacement rank, that is, where the rank of $\mathbf{A} - \mathbf{D}\mathbf{A}\mathbf{D}^T$ is low [49, 101–103]. Here \mathbf{D} is the “downshift” matrix, with ones on its first subdiagonal, and zeros elsewhere; the operation above subtracts

the largest leading principal submatrix of \mathbf{A} from its lower right corner. Toeplitz matrices are one example of matrices with low displacement rank; the displacement rank of a Toeplitz matrix is at most two. Other related matrices, such as inverses and products of Toeplitz matrices, may also be shown to have low displacement rank.

The displacement rank approach produces an algorithm for solving an equation with a matrix of displacement rank ρ in $\mathcal{O}(\rho M^2)$ operations. The covariance estimate of \mathbf{R} is a product of Toeplitz matrices, which has displacement rank 3; a matrix of the form $\mathbf{R} - \lambda \mathbf{I}$ has displacement rank 4, and $\mathbf{R} - \lambda \boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is Toeplitz, has displacement rank 5. By using displacement rank algorithms in eigenvalue computations similar to those performed here, it should be possible to develop $\mathcal{O}(NM^2)$ methods for finding N eigenvectors of a covariance matrix. This would remove the performance limitation caused by the use of a Toeplitz matrix.

Toeplitz Matrices as “Sparse” Matrices

A sparse matrix is one with few nonzero elements; from an algorithmic point of view, the distinguishing characteristics of a sparse matrix are that it requires much less than $\mathcal{O}(M^2)$ storage, and that matrix-vector multiplication requires less than $\mathcal{O}(M^2)$ operations. A wide variety of sparse matrix algorithms has been developed to take advantage of these characteristics. From our examination of Toeplitz matrices, it is clear that they share the algorithmic characteristics of sparse matrices: they require $\mathcal{O}(M)$ storage, and a matrix-vector multiplication may be performed in $\mathcal{O}(M \log M)$ operations. This suggests that existing sparse matrix algorithms may be well-suited for application to Toeplitz matrices as well.

For the frequency estimation problem, it is necessary to calculate a few of the largest or smallest eigenvalues and eigenvectors of a Toeplitz matrix. The Lanczos algorithm computes a tridiagonal matrix whose extremal eigenvalues usually converge to those of the input matrix in $\mathcal{O}(\sqrt{M})$ iterations [78], and each iteration requires

one matrix-vector multiplication. If this is performed with the FFT, each iteration requires $\mathcal{O}(M \log M)$ operations, and the overall algorithm is $\mathcal{O}(M^{3/2} \log M)$. This asymptotic complexity is less than the fast algorithms discussed here, but greater than the superfast algorithms; one potential advantage is that the only computation performed is matrix-vector multiplication, which is always stable. Like the algorithms developed here, the Lanczos algorithm also is easily extended to the generalized eigenvalue problem. In addition, since the fast methods for Toeplitz multiplication extend easily to the computation of the product of a covariance matrix and a vector, the Lanczos method provides another approach to a fast covariance eigensolver.

The use of the Lanczos algorithm to compute extremal eigenvalues of Toeplitz matrices was apparently first suggested by Ikramov [104]. In contrast to the Cybenko-Van Loan approach developed here, which has been pursued by several researchers, very little further study of the Lanczos approach has been performed; only a brief work by Huckle [105], and a recent paper by Xu and Kailath [106] have addressed this question.

Limits on Frequency Estimation

The Bhattacharyya bounds developed in Chapter 2 are the tightest known bounds for sinusoidal frequency estimation. They are still, however, smaller than the variance of all known frequency estimation algorithms in the threshold region, that is, for short data records, low signal-to-noise ratios, and closely spaced sinusoids. The fact that several different techniques, including exact maximum likelihood, MUSIC, and ESPRIT, all show very similar behavior in the threshold region is intriguing: it suggests two possibilities. First, it is possible that these apparently very different techniques share some common feature that limits their performance in the threshold region; in this case, identifying this feature might lead to estimation techniques with better threshold performance.

The second possibility is that the estimators' performance in the threshold region is limited by an as yet unknown bound. The second-order Bhattacharyya bounds developed here may be extended to higher orders, resulting in tighter bounds; although it is probably impractical to compute all the elements of the matrix which defines the third-order bound, it may be feasible to add selected higher order derivatives to the vector used to compute the second-order bound. If tighter bounds computed in this manner show that the performance of the best currently known frequency estimators is close to optimal, this would have strong implications for the development of future algorithms, shifting the focus of development away from increasing performance and toward enhancing efficiency.

In conclusion, it is pleasing to note that the approach described here for improving the efficiency of subspace frequency estimators appears not only to have useful application to situations where large amounts of data are available, but has also pointed the way to several new approaches for solving related problems. It is hoped that the pursuit of these new approaches is as fruitful as the investigation described here.

APPENDIX A REVIEW OF MATRIX ALGEBRA

In this appendix, a brief review of several important facts from matrix algebra will be given. To avoid the necessity of repeating the characteristics of the matrix for each fact, we will always assume that \mathbf{x} is a vector of length n , whose elements are denoted $x(i)$, and \mathbf{A} and \mathbf{B} are $n \times n$ matrices, whose entries are denoted $A(i, j)$ or $B(i, j)$.

Norms

Norms are used to measure the size of a matrix or vector. The most common vector norms are:

$$\begin{aligned}\|\mathbf{x}\|_1 &= \sum_{i=1}^n |x(i)|, \\ \|\mathbf{x}\|_2 &= \sqrt{\sum_{i=1}^n |x(i)|^2}, \\ \|\mathbf{x}\|_\infty &= \max_i |x(i)|.\end{aligned}$$

Each vector norm $\|\mathbf{x}\|_\alpha$ can also be used to define a matrix norm:

$$\|\mathbf{A}\|_\alpha = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{Ax}\|_\alpha}{\|\mathbf{x}\|_\alpha}.$$

In some cases, a matrix norm may be easily computed from the matrix itself:

$$\|\mathbf{A}\|_1 = \max_j \left\{ \sum_{i=1}^n |A(i, j)| \right\}$$

$$\begin{aligned}\|\mathbf{A}\|_{\infty} &= \max_i \left\{ \sum_{j=1}^n |A(i, j)| \right\} \\ \|\mathbf{A}\|_F &= \sqrt{\sum_{i=1}^n \sum_{j=1}^n |A(i, j)|^2}\end{aligned}$$

The norm $\|\mathbf{A}\|_F$ is the Frobenius norm, and is not derived from a vector norm. The norm $\|\mathbf{A}\|_2$ does not have a simple expression in terms of the elements of \mathbf{A} .

A norm $\|\mathbf{A}\|_{\beta}$ is *unitarily invariant* if, for any unitary matrices \mathbf{U} and \mathbf{V} of the proper dimensions,

$$\|\mathbf{UAV}\|_{\beta} = \|\mathbf{A}\|_{\beta}.$$

Special Matrices

A matrix is *Hermitian* if, for every element, $A(i, j) = A(j, i)^*$, where $*$ denotes complex conjugation; a matrix is *symmetric* if all of its entries are real and $A(i, j) = A(j, i)$. All symmetric matrices are Hermitian.

A Hermitian matrix is *positive definite* if, for all nonzero vectors \mathbf{x} , $\mathbf{x}^H \mathbf{A} \mathbf{x} > 0$; it is *positive semidefinite* if $\mathbf{x}^H \mathbf{A} \mathbf{x} \geq 0$ for all \mathbf{x} . All of the eigenvalues and all of the diagonal entries of a positive definite (positive semidefinite) matrix are greater than (greater than or equal to) zero.

A *Toeplitz* matrix is one for which $A(i, j) = A(k, l)$ whenever $i - j = k - l$. A Toeplitz matrix is completely determined by its first row and its first column.

A *unitary* matrix is one for which

$$\mathbf{A}^H = \mathbf{A}^{-1}.$$

Matrix Eigendecomposition

The *eigenvalues* λ_i of a matrix are the roots of the characteristic polynomial $\det(\lambda\mathbf{I} - \mathbf{A})$; there are at most n distinct eigenvalues. The *eigenvectors* of a matrix are the nonzero vectors \mathbf{e}_i that satisfy

$$\mathbf{A}\mathbf{e}_i = \lambda_i \mathbf{e}_i,$$

where λ_i is an eigenvalue of \mathbf{A} ; there are at most n linearly independent eigenvectors.

The eigenvectors of a matrix are not unique, since if \mathbf{e}_i is an eigenvector, then for any nonzero constant C , $C\mathbf{e}_i$ is also an eigenvector. By convention, this ambiguity is resolved by normalizing each eigenvector so that its length is one.

An *invariant subspace* of a matrix \mathbf{A} is a space \mathcal{S} such that if $\mathbf{x} \in \mathcal{S}$, then $\mathbf{Ax} \in \mathcal{S}$. An eigenvector defines a one-dimensional invariant subspace.

If \mathbf{A} has repeated eigenvalues, that is, if its characteristic polynomial has repeated roots, then the normalized eigenvectors associated with the repeated eigenvalues are not unique, since if \mathbf{e}_i and \mathbf{e}_j are eigenvectors associated with a repeated eigenvalue, then any (normalized) linear combination of \mathbf{e}_i and \mathbf{e}_j is also an eigenvector.

If \mathbf{A} is Hermitian, then all of its eigenvalues are real; if \mathbf{A} is symmetric, then both its eigenvalues and eigenvectors are real. In either case, n orthonormal eigenvectors of \mathbf{A} may always be found; but the eigenvectors associated with repeated eigenvalues are not uniquely determined.

The eigenvalues of a matrix may be shifted by any value α , without changing the eigenvectors, by adding a multiple of \mathbf{I} to the matrix, since

$$(\mathbf{A} + \alpha\mathbf{I})\mathbf{e}_i = (\lambda_i + \alpha)\mathbf{e}_i.$$

The product of the eigenvalues of a matrix is equal to its determinant, and the sum of the eigenvalues is equal to its trace, that is, to the sum of its diagonal entries.

If \mathbf{X} is $n \times n$ and nonsingular, then \mathbf{A} and $\mathbf{X}^{-1}\mathbf{A}\mathbf{X}$ have the same eigenvalues; \mathbf{A} and $\mathbf{X}^{-1}\mathbf{A}\mathbf{X}$ are *similar*.

The *inertia* of a Hermitian matrix is the number of positive, zero, and negative eigenvalues; if \mathbf{X} is $n \times n$ and nonsingular, then \mathbf{A} and $\mathbf{X}^H\mathbf{A}\mathbf{X}$ have the same inertia.

All of the eigenvalues of \mathbf{A} lie in the regions defined by

$$|A(i, i) - \lambda| \leq \sum_{j \neq i} |A(i, j)|;$$

this is known as Gershgorin's circle theorem.

If \mathbf{A} is Hermitian, $\tilde{\mathbf{A}}$ is a $n - 1 \times n - 1$ principal submatrix of \mathbf{A} , and the eigenvalues of $\tilde{\mathbf{A}}$ and \mathbf{A} are $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{n-1}$ and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$, respectively, then $\lambda_1 \geq \sigma_1 \geq \lambda_2 \geq \sigma_2 \geq \dots \lambda_{n-1} \geq \sigma_{n-1} \geq \lambda_n$.

Generalized Eigendecomposition

The *generalized eigenvalues* of a pair of matrices \mathbf{A} and \mathbf{B} are the n roots of the polynomial $\det(\lambda\mathbf{B} - \mathbf{A})$. A pair of matrices (\mathbf{A}, \mathbf{B}) is often referred to as a *matrix pencil*.

The *generalized eigenvectors* of the matrix pencil (\mathbf{A}, \mathbf{B}) are the nonzero vectors \mathbf{e}_i satisfying

$$\mathbf{A}\mathbf{e}_i = \lambda_i \mathbf{B}\mathbf{e}_i,$$

where λ_i is a generalized eigenvalue.

A *generalized invariant subspace* (sometimes called a *deflating subspace*) of a pencil (\mathbf{A}, \mathbf{B}) is a space \mathcal{S} such that if $\mathbf{B}\mathbf{x} \in \mathcal{S}$, then $\mathbf{A}\mathbf{x} \in \mathcal{S}$. A generalized eigenvector defines a one-dimensional generalized invariant subspace.

If both \mathbf{A} and \mathbf{B} are Hermitian, and there exists a matrix $\mathbf{C} = \alpha\mathbf{A} + (1 - \alpha)\mathbf{B}$ that is positive definite for some α between 0 and 1, inclusive, then the pencil (\mathbf{A}, \mathbf{B}) is *definite*. Notice that the pencil is definite if either \mathbf{A} or \mathbf{B} is positive definite.

The generalized eigenvalues of a definite pencil are real, and n generalized eigenvectors may always be found that are linearly independent and orthogonal with respect to the inner product defined by \mathbf{B} , that is,

$$\mathbf{e}_i^T \mathbf{B} \mathbf{e}_j = \delta_{ij}.$$

APPENDIX B ROUTINES FOR FAST SOLUTION OF TOEPLITZ SYSTEMS

Introduction

This section contains Fortran-90 subroutines which implement a superfast algorithm for the solution of Toeplitz systems. The subroutines employ a variant of the algorithm described by Ammar and Gragg [45]. The algorithm has been modified in the following ways: the sign of the input a has been reversed, and the forward Fourier transform is used in place of the inverse Fourier transform, and vice versa. The algorithm presented here computes the solution of $\mathbf{T}_n \mathbf{a} = -\mathbf{t}$, where \mathbf{T}_n is a $n \times n$ symmetric Toeplitz matrix, with $n = 2^k$, using $8n \log^2 n + 8n \log n + \mathcal{O}(n)$ real arithmetic operations.

The only features of the Fortran-90 language used in these subroutines which are not also legal in Fortran-77 are the `do ... enddo` construct, and recursion. Both of these features are widely supported extensions to the Fortran-77 language, so these subroutines will compile properly on many Fortran-77 systems as well.

These subroutines, and those in the other appendices which contain computer programs, were written using the `WEB` programming system [107]. This system allows both the source code and documentation to be written into a single combined source file, which is processed to generate the documentation and the program source code. This ensures that the documentation matches the program exactly.

Listings

1. Subroutine *ammar* is a driver routine that has the same parameters as the Levinson-Durbin algorithm: *r* is the autocorrelation, *a* the solution, *e* the prediction errors, and *n* the size of the problem. Note that the *n* input to *ammar* is the length of the autocorrelation sequence, which is one greater than the size of the matrix to be solved.

The WEB system provides a macro facility similar to the C language preprocessor's **#define**. This remedies a serious defect in the Fortran language. We will use the macro facility to define the largest allowable matrix; this value is needed to properly dimension arrays in several program units. The constant *NMAX* is the maximum value of *n*.

```
@m NMAX 214
```

2. The macro facility will also be used to simplify changing the precision of the program. A single precision version may be produced by redefining *FLOAT* to **real**, *FLT(x)* to **real(x)**, and the constants to real numbers.

```
@m FLOAT double precision
@m FLT(x) dble(x)
@m ZERO 0. · 100D
@m ONE 1. · 100D
```

3. The subroutine *ammar* itself checks the validity of its arguments, performs the generalized Schur algorithm, and returns the solution to the Yule-Walker problem $\mathbf{T}\mathbf{a} = -\mathbf{t}$ in *a*. If the solution to the general symmetric Toeplitz system $\mathbf{T}\mathbf{x} = \mathbf{b}$ is required, *a* may be used in the Gohberg-Semencul formula to calculate \mathbf{T}^{-1} , as described in Chapter 5.

```
subroutine ammar(r,a,e,n)
  integer n
  FLOAT r(n+1),a(n),e(n+1)
  {Declare ammar's local variables. 4}
  {Check that n is valid. 5}
  {Solve the Yule-Walker problem. 6}
  return
end
```

4. The only local variables used are u and v , which store the Fourier transforms of the vectors produced by the Schur algorithm. The algorithm as a whole requires $O(NMAX)$ storage.

\langle Declare *ammar*'s local variables. 4 $\rangle \equiv$

```
integer iex, i
FLOAT u(NMAX),v(NMAX)
```

This code is used in section 3.

5. The input value of n is checked for validity.

\langle Check that n is valid. 5 $\rangle \equiv$

```
iex = nint(log(FLT(n))/log(FLT(2)))
if (2iex  $\neq$  n) then
    stop 'Error: n must be a power of 2 in ammar().'
endif
if (n  $\leq$  1) then
    stop 'Error: n is less than 2 in ammar().'
endif
if (n > NMAX) then
    stop 'Error: n is too large in ammar().'
endif
```

This code is used in section 3.

6. The generalized Schur algorithm is used to solve the Yule-Walker equations. One of the differences between this implementation and that of Ammar and Gragg is that the sign of a has been reversed. This allows the input vector r to be used directly as input to the subroutine that implements the algorithm.

In this section and those below, a matrix notation will be used to describe the calculations. The matrices \mathbf{F}_n and \mathbf{F}_n^{-1} represent the forward and inverse discrete Fourier transforms, respectively; \mathbf{S}_n and \mathbf{C}_n represent noncyclic and cyclic downward shift matrices, and $[\mathbf{a}, \mathbf{b}]$ represents a matrix formed from the column vectors \mathbf{a} and \mathbf{b} . In this notation, the first portion of *ammar* performs the operation

$$\mathbf{a} = \mathbf{F}_n^{-1}\mathbf{u} + \mathbf{S}\mathbf{F}_n^{-1}\mathbf{v}.$$

These operations could be performed as shown above, however, one inverse FFT can be saved by combining the vectors in the transform domain. This is possible because the first coefficient of $\mathbf{F}_n^{-1}\mathbf{v}$ is always one. If the Fourier transform of a vector with one as its first entry and zeros elsewhere is subtracted from v , the resulting vector may

now be circularly shifted to yield the desired result. These operations are performed by the subroutine *fixv*.

The last step is the calculation of the final prediction error $e(n)$.

(Solve the Yule-Walker problem. 6) \equiv

```

call gsa(r(2),r(1),u,v,e,n)
call fixv(v,n)
do i = 1,n
  a(i) = u(i) + v(i)
enddo
call irfft(a,n)
e(n + 1) = (ONE - a(n))*(ONE + a(n))*e(n)

```

This code is used in section 3.

7. Subroutine *gsa* is the heart of the implementation; it is the only recursive routine required. The keyword **recursive** is required by standard Fortran-90; it may also be required by some Fortran-77 compilers.

Since the ordinary (nonrecursive) Schur algorithm is more efficient for small n , it is used once the problem size falls below a threshold. The size at which the recursive (split) algorithm becomes faster is given by the macro *NSPLIT*.

```

@m NSPLIT 64
recursive subroutine gsa(a,b,u,v,e,n)
  integer n
  FLOAT a(n),b(n),u(n),v(n),e(n + 1)
  (Declare gsa's local variables. 8)
  if (n ≤ NSPLIT) then
    call schur(a,b,u,v,e,n)
  else
    (Solve using the generalized Schur algorithm. 9)
  endif
  return
end

```

8. The subroutine *gsa* requires local storage proportional to *NMAX*. Since the subroutine is recursive, the program must be designed so that local variables do not overwrite themselves on successive calls. There are three general methods to ensure this. The first is to allocate local variables on the program stack; this requires a very large stack, and it is generally impossible for the program to determine if enough stack

space is available at run time. The second option is to dynamically allocate memory; this is standard if Fortran-90, but there is no standard syntax for it even in those Fortran-77 compilers that support dynamic allocation. The third option is to use standard statically allocated arrays, and a slightly more complex indexing scheme, to ensure that each level of recursive execution uses different memory locations. This is the approach adopted here.

The arrays that are reused by recursive calls are first listed in a **save** statement, to prevent Fortran-90 compilers from allocating them on the stack. Then, each array is declared to be twice the size required by the highest level subroutine. On each recursive call, for a workspace size of k , the memory locations $k + 1$ to $2k$ are used, ensuring that each level of recursion uses different memory locations. For example, if the number of memory locations required was 8, the first level subroutine call would use locations 9 through 16, the second level recursive calls would use locations 5 through 8, and so on.

(Declare gsa's local variables. 8) \equiv

```
save p, q, r, s, c, d, p1, q1, a1, b1, t1, t2, x1, y1, u1, v1
integer i, m, np1, mp1
FLOATTp(2*NMAX),q(2*NMAX),r(2*NMAX),s(2*NMAX)
FLOATx1(NMAX/2),y1(NMAX/2),u1(NMAX),v1(NMAX)
FLOATc(NMAX),d(NMAX),p1(2*NMAX),q1(2*NMAX)
FLOATa1(NMAX),b1(NMAX),t1(NMAX),t2(NMAX)
```

This code is used in section 7.

9. Each step of the generalized Schur algorithm makes two recursive calls to solve problems of half the size, and performs $O(n \log n)$ additional computation, leading to an overall $O(n \log^2 n)$ computational load. The symbols used for the variables are those used in [45] with two exceptions, noted below.

(Solve using the generalized Schur algorithm. 9) \equiv

```
m = n/2
mp1 = m + 1
np1 = n + 1
call gsa(a,b,u1(mp1),v1(mp1),e,m)
  ( Compute p and q. 10 )
  ( Compute r and s. 11 )
  ( Compute c and d. 12 )
  ( Compute a1 and b1. 13 )
call gsa(a1(mp1),b1(mp1),u1(mp1),v1(mp1),e(mp1),m)
  ( Compute p1 and q1. 14 )
  ( Compute u and v. 15 )
```

This code is used in section 7.

10. After the first recursive call, the outputs of the previous stage are used to calculate the inputs to the second call. In the first step below, the vectors \mathbf{x}_1 and \mathbf{y}_1 are calculated:

$$[\mathbf{x}_1, \mathbf{y}_1] = \mathbf{F}_{n/2}^{-1}[\mathbf{u}_1, \mathbf{v}_1].$$

Since \mathbf{u}_1 and \mathbf{v}_1 are the Fourier transforms of real vectors, a real split-radix IFFT may be used.

In the original implementation [45] the output vectors were \mathbf{x}_0 and \mathbf{y}_0 , but since the two sets of vectors are never needed simultaneously, the same array may be shared with the \mathbf{x}_1 and \mathbf{y}_1 needed later.

The second step calculates

$$[\mathbf{p}, \mathbf{q}] = \mathbf{F}_n \begin{bmatrix} \mathbf{x}_1 & \mathbf{y}_1 \\ 0 & 0 \end{bmatrix}.$$

This appears to require two length n real-valued FFTs, but since the length $n/2$ Fourier coefficients are already known, Ammar and Gragg recognized that each of the two length n FFTs could be calculated using only one additional complex FFT of length $n/4$. This calculation is performed by subroutine *zpcal*.

\langle Compute \mathbf{p} and \mathbf{q} . 10 $\rangle \equiv$

```

do i = 1, m
  x1(i) = u1(m + i)
  y1(i) = v1(m + i)
enddo
call ifft(x1, m)
call ifft(y1, m)
call zpcal(x1, u1(mp1), p(np1), n)
call zpcal(y1, v1(mp1), q(np1), n)

```

This code is used in section 9.

11. The values of \mathbf{r} and \mathbf{s} are given by

$$[\mathbf{r}, \mathbf{s}] = \mathbf{F}_n \mathbf{E}_n \begin{bmatrix} \mathbf{J}_m \mathbf{x}_1 & \mathbf{J}_m \mathbf{y}_1 \\ 0 & 0 \end{bmatrix}.$$

These values may be calculated by simply rearranging \mathbf{p} and \mathbf{q} , so no additional operations are required. The rearrangement is performed by subroutine *flip*.

\langle Compute \mathbf{r} and \mathbf{s} . 11 $\rangle \equiv$

```
call flip(p(np1), r(np1), n)
```

```
call flip(q(np1),s(np1),n)
```

This code is used in section 9.

12. The vectors **c** and **d** are just the Fourier transforms of the input vectors **a** and **b**, adjusted to take account of the fact that the sign of **a** has been reversed.

$$[\mathbf{c}, \mathbf{d}] = \mathbf{F}_n[-\mathbf{a}, \mathbf{b}]$$

$\langle \text{Compute } \mathbf{c} \text{ and } \mathbf{d}. \text{ 12} \rangle \equiv$

```
do i = 1, n
  c(i) = -a(i)
  d(i) = b(i)
enddo
call rfft(c,n)
call rfft(d,n)
```

This code is used in section 9.

13. The vectors **a₁** and **b₁** are the inputs to the second recursive call of *gsa*.

$$\begin{bmatrix} \times \\ \mathbf{a}_1 \end{bmatrix} = \mathbf{F}_n^{-1}[\mathbf{d} \cdot \mathbf{p} - \mathbf{c} \cdot \mathbf{q}]$$

$$\begin{bmatrix} \times \\ \mathbf{b}_1 \end{bmatrix} = \mathbf{F}_n^{-1}[\mathbf{d} \cdot \mathbf{s} - \mathbf{c} \cdot \mathbf{r}]$$

$\langle \text{Compute } \mathbf{a}_1 \text{ and } \mathbf{b}_1. \text{ 13} \rangle \equiv$

```
call fftmul(c,q(np1),t1,n)
call fftmul(d,p(np1),t2,n)
do i = 1, n
  t1(i) = t2(i) - t1(i)
enddo
call irfft(t1,n)
do i = m + 1, n
  a1(i) = t1(i)
enddo
call fftmul(d,s(np1),t1,n)
call fftmul(c,r(np1),t2,n)
do i = 1, n
  t1(i) = t1(i) - t2(i)
enddo
```

```

call irfft(t1,n)
do i = m + 1, n
  b1(i) = t1(i)
enddo

```

This code is used in section 9.

14. The computation of \mathbf{p}_1 and \mathbf{q}_1 takes advantage of the same “trick” used to compute \mathbf{p} and \mathbf{q} .

$$[\mathbf{p}_1, \mathbf{q}_1] = \mathbf{F}_n^{-1} \begin{bmatrix} \mathbf{x}_1 & \mathbf{y}_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

\langle Compute \mathbf{p}_1 and \mathbf{q}_1 . 14 $\rangle \equiv$

```

do i = 1, m
  x1(i) = u1(m + i)
  y1(i) = v1(m + i)
enddo
call irfft(x1,m)
call irfft(y1,m)
call zpcal(x1,u1(mp1),p1(np1),n)
call zpcal(y1,v1(mp1),q1(np1),n)

```

This code is used in section 9.

15. Finally, the Fourier transforms of the results are computed.

$$\begin{aligned} \mathbf{u} &= [\mathbf{s} \cdot \mathbf{p}_1 + \mathbf{p} \cdot \mathbf{q}_1] \\ \mathbf{v} &= [\mathbf{r} \cdot \mathbf{p}_1 + \mathbf{q} \cdot \mathbf{q}_1] \end{aligned}$$

\langle Compute \mathbf{u} and \mathbf{v} . 15 $\rangle \equiv$

```

call fftmul(s(np1),p1(np1),t1,n)
call fftmul(p(np1),q1(np1),t2,n)
do i = 1, n
  u(i) = t1(i) + t2(i)
enddo
call fftmul(r(np1),p1(np1),t1,n)
call fftmul(q(np1),q1(np1),t2,n)
do i = 1, n
  v(i) = t1(i) + t2(i)
enddo

```

This code is used in section 9.

16. The normal Schur algorithm is used for $n \leq \text{NSPLIT}$.

```

subroutine schur(a,b,u,v,e,n)
  integer n
  FLOAT a(n), b(n), u(0 : n - 1), v(0 : n - 1), e(n + 1)
  ( allocate schur's local variables and initialize 17 )
  do k = 1, n
    ( calculate alphak and betak 18 )
    ( calculate uk and vk 19 )
  enddo
  e(n + 1) = be(0)
  call rfft(u,n)
  call rfft(v,n)
  return
end

```

17. *(allocate schur's local variables and initialize 17)* \equiv

```

  integer k, j, kmj
  FLOAT xk(NSPLIT)
  FLOAT al(0 : NSPLIT - 1), be(0 : NSPLIT - 1)
  FLOAT t0(0 : NSPLIT - 1), t1(0 : NSPLIT - 1)
  do j = 1, n - 1
    u(j) = ZERO
    v(j) = ZERO
  enddo
  u(0) = ZERO
  v(0) = ONE
  xk(1) = ZERO

```

This code is used in section 16.

18. *(calculate alpha_k and beta_k 18)* \equiv

```

  al(k - 1) = -a(k)
  be(k - 1) = b(k)
  do j = 1, k - 1
    kmj = k - j
    al(kmj - 1) = al(kmj) - xk(j)*be(kmj)
    be(kmj) = be(kmj) - xk(j)*al(kmj)
  enddo
  xk(k) = al(0)/be(0)
  e(k) = be(0)
  be(0) = be(0)*(ONE - xk(k))*(ONE + xk(k))

```

This code is used in section 16.

19. \langle calculate u_k and v_k \rangle \equiv

```
do j = 0, k - 1
    t0(j) = u(j)
    t1(j) = v(j)
enddo
do j = 0, k - 1
    kmj = k - j - 1
    u(j) = xk(k)*t1(kmj) + t0(j)
    v(j) = xk(k)*t0(kmj) + t1(j)
enddo
```

This code is used in section 16.

REFERENCES

- [1] D. C. Rife and R. R. Boorstyn, "Single-tone parameter estimation from discrete-time observations", *IEEE Transactions on Information Theory*, vol. 20, no. 5, pp. 591–598, Sept. 1974.
- [2] P. Stoica and A. Nehorai, "MUSIC, maximum likelihood and Cramér-Rao bound", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 5, pp. 720–741, May 1989.
- [3] P. Stoica, R. L. Moses, B. Friedlander, and T. Sönderström, "Maximum likelihood estimation of the parameters of multiple sinusoids from noisy measurements", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 378–392, Mar. 1989.
- [4] B. Porat, *Digital Processing of Random Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.
- [5] C. R. Rao, "Information and the accuracy attainable in the estimation of statistical parameters", *Bulletin of the Calcutta Mathematical Society*, vol. 37, pp. 81–91, 1945.
- [6] A. Bhattacharyya, "On some analogues of the amount of information and their use in statistical parameter estimation", *Sankhyā*, vol. 8, pp. 1–14, 201–218, 315–328 (3 parts), 1947–48.
- [7] D. C. Rife and R. R. Boorstyn, "Multiple tone parameter estimation from discrete-time observations", *Bell System Technical Journal*, vol. 55, no. 9, pp. 1389–1410, Nov. 1976.
- [8] P. Stoica and A. Nehorai, "MUSIC, maximum likelihood and Cramér-Rao bound: Further results and comparisons", in *Proceedings of ICASSP '89*, 1989, pp. 2605–2608.
- [9] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, PTR Prentice-Hall, Englewood Cliffs, New Jersey, 1993.
- [10] R. O. Schmidt, *A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation*, Ph. D. dissertation, Stanford University, Stanford, California, 1982.
- [11] H. Akaike, "A new look at the statistical model identification", *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–722, Dec. 1974.

- [12] G. Schwartz, "Estimating the dimension of a model", *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.
- [13] J. Rissanen, "Modeling by shortest data description", *Automatica*, vol. 14, pp. 465–471, 1978.
- [14] J. Rissanen, "A universal prior for the integers and estimation by minimum description length", *Annals of Statistics*, vol. 11, no. 2, pp. 416–431, 1983.
- [15] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 387–392, Apr. 1985.
- [16] M. Wax, *Detection and Estimation of Superimposed Signals*, Ph. D. dissertation, Stanford University, Stanford, California, 1985.
- [17] D. H. Johnson and D. E. Dudgeon, *Array Signal Processing*, PTR Prentice-Hall, Englewood Cliffs, New Jersey, 1993.
- [18] M. Wax and I. Ziskind, "Detection of the number of coherent signals by the MDL principle", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 8, pp. 1190–1196, Aug. 1989.
- [19] M. Wax and I. Ziskind, "Detection and localization of multiple sources via the stochastic signals model", *IEEE Transactions on Signal Processing*, vol. 39, no. 11, pp. 2450–2456, Nov. 1991.
- [20] V. Pisarenko, "The retrieval of harmonics from a covariance function", *Geophysical Journal of the Royal Astronomical Society*, vol. 33, pp. 347–366, 1973.
- [21] P. Stoica and A. Nehorai, "Study of the statistical performance of the Pisarenko harmonic decomposition method", *IEE Proceedings, Part F*, vol. 135, no. 2, pp. 161–168, Apr. 1988.
- [22] D. W. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood", *Proceedings of the IEEE*, vol. 70, no. 9, pp. 975–989, Sept. 1982.
- [23] R. Kumaresan, *Estimating the Parameters of Exponentially Damped or Undamped Sinusoidal Signals in Noise*, Ph. D. dissertation, University of Rhode Island, Kingston, Rhode Island, 1982.
- [24] D. W. Tufts and R. Kumaresan, "Singular value decomposition and improved frequency estimation using linear prediction", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 30, no. 4, pp. 671–675, Aug. 1982.

- [25] H. Clergeot, S. Tressens, and A. Ouamri, “Performance of high resolution frequencies estimation methods compared to the Cramér-Rao bounds”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1703–1720, Nov. 1989.
- [26] R. O. Schmidt, “Multiple emitter location and signal parameter estimation”, *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, 1986.
- [27] A. J. Barabell, “Improving the resolution performance of eigenstructure-based direction-finding algorithms”, in *Proceedings of ICASSP '83*, 1983, pp. 336–339.
- [28] B. D. Rao and K. V. S. Hari, “Performance analysis of root-MUSIC”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1939–1949, Dec. 1989.
- [29] P. Stoica and A. Nehorai, “Statistical analysis of MUSIC and subspace rotation estimates of sinusoidal frequencies”, *IEEE Transactions on Signal Processing*, vol. 39, no. 8, pp. 1836–1847, Aug. 1991.
- [30] M. Kaveh and A. J. Barabell, “The statistical performance of the MUSIC and minimum-norm algorithms in resolving plane waves in noise”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 331–341, Apr. 1986, (see also the corrections in vol. 34, p. 633).
- [31] X.-L. Xu and K. M. Buckley, “Bias analysis of the MUSIC location estimator”, *IEEE Transactions on Signal Processing*, vol. 40, no. 10, pp. 2599–2569, Oct. 1992.
- [32] P. Stoica and A. Nehorai, “Performance comparison of subspace rotation and MUSIC methods for direction estimation”, *IEEE Transactions on Signal Processing*, vol. 39, no. 2, pp. 446–453, Feb. 1991.
- [33] R. H. Roy, A. Paulraj, and T. Kailath, “ESPRIT–A subspace rotation approach to estimation of parameters of cisooids in noise”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1340–1342, Oct. 1986.
- [34] R. H. Roy, *ESPRIT–Estimation of Signal Parameters via Rotational Invariance Techniques*, Ph. D. dissertation, Stanford University, Stanford, California, 1987.
- [35] R. H. Roy and T. Kailath, “ESPRIT–Estimation of signal parameters via rotational invariance techniques”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984–995, July 1989.
- [36] A. L. Swindlehurst, B. Ottersten, R. H. Roy, and T. Kailath, “Multiple invariance ESPRIT”, *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 867–881, Apr. 1992.

- [37] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 2nd edition, 1989.
- [38] B. Ottersten, M. Viberg, and T. Kailath, “Performance analysis of the total least squares ESPRIT algorithm”, *IEEE Transactions on Signal Processing*, vol. 39, no. 5, pp. 1122–1135, May 1991.
- [39] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, John Wiley and Sons, New York, 1984.
- [40] C. Davis and W. M. Kahan, “The rotation of eigenvectors by a perturbation, III”, *SIAM Journal of Numerical Analysis*, vol. 7, pp. 1–46, Mar. 1970.
- [41] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, San Diego, California, 1990.
- [42] R. R. Bitmead and B. D. O. Anderson, “Asymptotically fast solution of Toeplitz and related systems of linear equations”, *Linear Algebra and its Applications*, vol. 34, pp. 103–116, 1980.
- [43] R. P. Brent, F. G. Gustafson, and D. Y. Y. Yun, “Fast solution of Toeplitz systems of equations and computation of Padé approximants”, *Journal of Algorithms*, vol. 1, pp. 259–295, 1980.
- [44] F. de Hoog, “A new algorithm for solving Toeplitz systems of equations”, *Linear Algebra and its Applications*, vol. 88–89, pp. 123–138, 1987.
- [45] G. S. Ammar and W. B. Gragg, “Superfast solution of real positive definite Toeplitz systems”, *SIAM Journal of Matrix Analysis and Applications*, vol. 9, no. 1, pp. 61–76, Jan. 1988.
- [46] T. Kailath, A. Viera, and M. Morf, “Inverses of Toeplitz operators, innovations, and orthogonal polynomials”, *SIAM Review*, vol. 20, pp. 106–118, Jan. 1987.
- [47] S. L. Marple, *Digital Spectral Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.
- [48] T. Kailath, “A theorem of I. Schur and its impact on modern signal processing”, in *I. Schur Methods in Operator Theory and Signal Processing*, vol. 18 of *Operator Theory: Advances and Applications*, pp. 9–30. Birkhäuser Verlag, Basel, 1986.
- [49] H. Lev-Ari and T. Kailath, “Triangular factorization of structured Hermitian matrices”, in *I. Schur Methods in Operator Theory and Signal Processing*, vol. 18 of *Operator Theory: Advances and Applications*, pp. 301–324. Birkhäuser Verlag, Basel, 1986.

- [50] C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.
- [51] S.-Y. Kung and Y. H. Hu, “A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, pp. 66–76, Feb. 1983.
- [52] G. S. Ammar and W. B. Gragg, “The generalized Schur algorithm for the superfast solution of Toeplitz systems”, in *Rational Approximation and its Applications in Mathematics and Physics*, J. Gilewicz, M. Pindor, and W. Siemaszko, Eds., pp. 315–330. Springer-Verlag, Berlin, 1985.
- [53] G. S. Ammar and W. B. Gragg, “The implementation and use of the generalized Schur algorithm”, in *Computational and Combinatorial Methods in Systems Theory*, C. I. Byrnes and A. Lindquist, Eds., pp. 265–279. Elsevier Science Publishers, Amsterdam, 1986.
- [54] B. R. Musicus, “Levinson and fast Choleski algorithms for Toeplitz and almost Toeplitz matrices”, Technical Report 538, MIT Research Laboratory of Electronics, Cambridge, Massachusetts, 1988.
- [55] R. Kumar, “A fast algorithm for solving Toeplitz system of equations”, in *Proceedings of ICASSP '83*, 1983, pp. 166–169.
- [56] G. S. Ammar and W. B. Gragg, “Numerical experience with a real superfast Toeplitz solver”, *Linear Algebra and its Applications*, vol. 121, pp. 185–206, 1989.
- [57] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus, “Real-valued fast Fourier transform algorithms”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, pp. 849–863, June 1987.
- [58] H. V. Sorensen and C. S. Burrus, “Fast DFT and convolution algorithms”, in *Handbook for Digital Signal Processing*, S. K. Mitra and J. F. Kaiser, Eds., chapter 8. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.
- [59] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1992.
- [60] J. R. Jain, “An efficient algorithm for a large Toeplitz set of linear equations”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 6, pp. 612–615, Dec. 1979.
- [61] I. C. Gohberg and I. A. Fel'dman, *Convolution Equations and Projection Methods for their Solution*, American Mathematical Society, Providence, Rhode Island, 1974.

- [62] X. Zhong, “On Moore-Penrose inverses of Toeplitz matrices”, *Linear Algebra and its Applications*, vol. 169, pp. 9–15, 1992.
- [63] G. Heinig and F. Hellinger, “Displacement structure of pseudoinverses”, *Linear Algebra and its Applications*, vol. 197–198, pp. 623–649, 1994.
- [64] J. R. Bunch, “The weak and strong stability of algorithms in numerical linear algebra”, *Linear Algebra and its Applications*, vol. 88–89, pp. 49–66, 1987.
- [65] G. Cybenko, “The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations”, *SIAM Journal of Scientific and Statistical Computing*, vol. 1, pp. 303–319, Sept. 1980.
- [66] J. R. Bunch, “Stability of methods for solving Toeplitz systems of equations”, *SIAM Journal of Scientific and Statistical Computing*, vol. 6, no. 2, pp. 349–364, Apr. 1985.
- [67] F. T. Luk and S. Qiao, “A fast but unstable orthogonal factorization technique for Toeplitz matrices”, *Linear Algebra and its Applications*, vol. 88–89, pp. 495–506, 1987.
- [68] T. F. Chan and P. C. Hansen, “A look-ahead Levinson algorithm for indefinite Toeplitz systems”, *SIAM Journal of Matrix Analysis and Applications*, vol. 13, no. 2, pp. 490–506, Apr. 1992.
- [69] C. J. Zarowski, “Schur algorithms for Hermitian Toeplitz, and Hankel matrices with singular leading principal submatrices”, *IEEE Transactions on Signal Processing*, vol. 39, pp. 2464–2480, Nov. 1991.
- [70] I. Koltracht and P. Lancaster, “Condition numbers of Toeplitz and block Toeplitz matrices”, in *I. Schur Methods in Operator Theory and Signal Processing*, vol. 18 of *Operator Theory: Advances and Applications*, pp. 271–300. Birkhäuser Verlag, Basel, 1986.
- [71] G. Cybenko and C. F. Van Loan, “Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix”, *SIAM Journal of Scientific and Statistical Computing*, vol. 7, no. 1, pp. 123–131, Jan. 1986.
- [72] M. H. Hayes and M. A. Clements, “An efficient algorithm for computing Pisarenko’s harmonic decomposition using Levinson’s recursion”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 485–491, June 1986.
- [73] Y. H. Hu and S.-Y. Kung, “Toeplitz eigensystem solver”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 1264–1271, Oct. 1985.

- [74] W. Barth, R. S. Martin, and J. H. Wilkinson, “Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection”, *Numerische Mathematik*, vol. 9, pp. 386–393, 1967.
- [75] W. F. Trench, “Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices”, *SIAM Journal of Matrix Analysis and Applications*, vol. 10, no. 2, pp. 135–146, Apr. 1989.
- [76] W. F. Trench, “Numerical solution of the eigenvalue problem for efficiently structured Hermitian matrices”, *Linear Algebra and its Applications*, vol. 154–156, pp. 415–432, 1991.
- [77] F. Noor and S. D. Morgera, “Recursive and iterative algorithms for computing eigenvalues of Hermitian Toeplitz matrices”, *IEEE Transactions on Signal Processing*, vol. 41, pp. 1272–1280, Mar. 1993.
- [78] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.
- [79] G. Feyh and C. T. Mullis, “Inverse eigenvalue problem for real symmetric Toeplitz matrices”, in *Proceedings of ICASSP '88*, 1988, pp. 1636–1639.
- [80] D. Hertz, “Simple bounds on the extreme eigenvalues of Toeplitz matrices”, *IEEE Transactions on Signal Processing*, vol. 40, pp. 175–176, Jan. 1992.
- [81] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.
- [82] B. N. Parlett, “The Rayleigh quotient iteration and some generalizations for nonnormal matrices”, *Mathematics of Computation*, vol. 28, pp. 679–693, July 1974.
- [83] F. M. Larkin, “Root-finding by fitting rational functions”, *Mathematics of Computation*, vol. 35, no. 151, pp. 803–816, July 1980.
- [84] V. Norton, “Algorithm 631: Finding a bracketed zero by Larkin’s method of rational interpolation”, *ACM Transactions on Mathematical Software*, vol. 11, pp. 120–134, June 1985.
- [85] P. C. Hansen and T. F. Chan, “Fortran subroutines for general Toeplitz systems”, *ACM Transactions on Mathematical Software*, vol. 18, no. 3, pp. 256–273, Sept. 1992.
- [86] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.

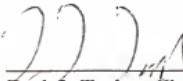
- [87] A. Cantoni and P. Butler, “Eigenvalues and eigenvectors of symmetric centrosymmetric matrices”, *Linear Algebra and its Applications*, vol. 13, pp. 275–288, 1976.
- [88] W. F. Trench, “Interlacement of the even and odd spectra of real symmetric Toeplitz matrices”, *Linear Algebra and its Applications*, vol. 195, pp. 59–68, 1993.
- [89] T. F. Chan, “An improved algorithm for computing the singular value decomposition”, *ACM Transactions on Mathematical Software*, vol. 8, no. 1, pp. 72–83, Mar. 1982.
- [90] J.-G. Sun, “The perturbation bounds for eigenspaces of a definite matrix-pair”, *Numerische Mathematik*, vol. 41, pp. 321–343, 1983.
- [91] J.-G. Sun, “Perturbation analysis for the generalized eigenvalue and the generalized singular value problem”, in *Matrix Pencils*, B. Kågström and A. Ruhe, Eds., pp. 221–244. Springer-Verlag, Berlin, 1983.
- [92] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, San Diego, California, 1981.
- [93] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK User’s Guide*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1979.
- [94] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines – EISPACK Guide*, Springer-Verlag, Berlin, 1974.
- [95] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler, *Matrix Eigensystem Routines – EISPACK Guide Extension*, Springer-Verlag, Berlin, 1977.
- [96] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1991.
- [97] K. A. Gallivan, M. T. Heath, E. Ng, J. M. Ortega, B. W. Peyton, R. J. Plemmons, C. H. Romnie, A. H. Sameh, and R. G. Voigt, *Parallel Algorithms for Matrix Computations*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1990.
- [98] J. P. Burg, D. G. Luneberger, and D. L. Wenger, “Estimation of structured covariance matrices”, *Proceedings of the IEEE*, vol. 70, pp. 963–974, Sept. 1982.
- [99] D. R. Fuhrmann and M. I. Miller, “On the existence of positive-definite maximum-likelihood estimates of structured covariance matrices”, *IEEE Transactions on Information Theory*, vol. 34, no. 4, pp. 722–729, July 1988.

- [100] D. B. Williams and D. H. Johnson, "Robust estimation of structured covariance matrices", *IEEE Transactions on Signal Processing*, vol. 41, no. 9, pp. 2891–2906, Sept. 1993.
- [101] T. Kailath, S.-Y. Kung, and M. Morf, "Displacement ranks of matrices and linear equations", *Journal of Mathematical Analysis and Applications*, vol. 68, pp. 395–407, 1979.
- [102] B. Friedlander, M. Morf, T. Kailath, and L. Ljung, "New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices", *Linear Algebra and its Applications*, vol. 27, pp. 31–60, 1979.
- [103] I. Gohberg, T. Kailath, and I. Koltracht, "Efficient solution of linear equations with recursive structure", *Linear Algebra and its Applications*, vol. 80, 1986.
- [104] K. D. Ikramov, "The complexity of some spectral problems for Toeplitz matrices", *USSR Computational Mathematics and Mathematical Physics*, vol. 21, no. 4, pp. 216–221, 1981.
- [105] T. Huckle, "Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix with spectral transformation Lanczos methods", in *Numerical Treatment of Eigenvalue Problems*, J. Albrecht, L. Collatz, P. Hagedorn, and W. Vete, Eds., vol. 5, pp. 109–115. Birkhäuser Verlag, Basel, 1991.
- [106] G. Xu and T. Kailath, "Fast subspace decomposition", *IEEE Transactions on Signal Processing*, vol. 42, no. 3, pp. 539–551, Mar. 1994.
- [107] D. E. Knuth, *Literate Programming*, Center for the Study of Language and Information, Stanford University, Stanford, California, 1992.

BIOGRAPHICAL SKETCH

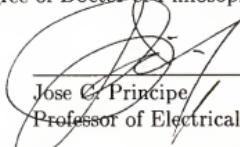
Tom Wallace was born and raised in Florida; he received the BEE. and the MS. in technology and science policy from Georgia Tech in 1983 and 1985. After working for several years in research and development in the defense industry, he returned to graduate school, receiving the MS. in electrical engineering from the University of Florida in 1992. Since 1987, he has been with ARCO Power Technologies, Inc., in Washington, D. C., where he is currently a Systems Engineer.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Fred J. Taylor, Chairman
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



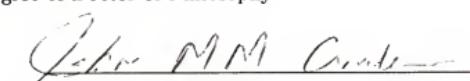
Jose C. Principe
Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



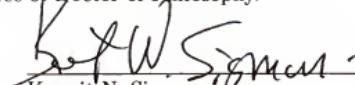
Scott L. Miller
Associate Professor of Electrical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



John M. M. Anderson
Assistant Professor of Electrical Engineering

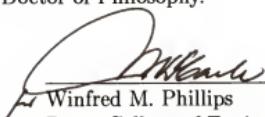
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Kermit N. Sigmon
Associate Professor of Mathematics

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August, 1994



Winfred M. Phillips
Dean, College of Engineering

Karen A. Holbrook
Dean, Graduate School